uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Informática

2017-2018

Trabajo Fin de Grado

Gestión Autónoma de Vuelo e Integración de Eventos de Misión en Plataforma Pixhawk

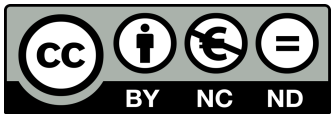
---

Enrique Martínez Martínez

Tutor

Jesús García Herrero

Leganés, 2018



RESUMEN

Los vehículos aéreos no tripulados, también llamados drones, son una tecnología que actualmente está en desarrollo, actualmente los drones todavía necesitan de un piloto controlando el dron de forma remota, aunque se están empezando a desarrollar pilotos automáticos para este tipo de aeronaves. Los drones pilotados son actualmente muy utilizados y están apareciendo muchos nuevos usos para estos vehículos, uno de los usos más comunes es como vehículo aéreo de grabación, tanto para uso particular como para uso profesional. Un ejemplo de un uso profesional para este tipo de aeronaves es la vigilancia del tráfico en una ciudad.

Este trabajo de fin de grado se centra en el desarrollo de un piloto automático para drones, concretamente la incorporación de un sistema de detección y evasión de obstáculos. Este proyecto está desarrollado para la plataforma PX4 que forma parte de un proyecto llamado DroneCode. DroneCode incorpora una serie de proyectos hardware y software que trabajan conjuntamente para el vuelo de drones con un piloto automático.

Este proyecto tiene como objetivo principal el tratamiento de eventos y análisis de la arquitectura de PX4. Se ha decidido añadir un sistema de evasión sencillo para que el objetivo principal de este proyecto tenga una utilidad y sea el comienzo de un proyecto donde se mejore este sistema de evasión para conseguir un piloto automático mucho más efectivo.

Aunque el proyecto principal se centra en PX4, en este proyecto también se abordarán tecnologías complementarias como, por ejemplo, Mavlink o Gazebo, herramientas integradas en el entorno DroneCode. Gazebo es el simulador que se utilizará en este proyecto.

Palabras clave

PX4, dron, sensor, arquitectura, evasión de obstáculos.

ABSTRACT

Unmanned aerial vehicles, also called drones, are a technology that is currently under development. Nowadays drones still need a pilot controlling the drone remotely, although it is beginning to be developed autopilots for this type of aircraft. Piloted drones are widely used and many new uses are appearing for these vehicles. One of the most common uses is as a recording aerial vehicle, both for private use and for professional use. An example of a professional use for this type of aircraft is monitoring of traffic in a city.

This final project focuses on the development of an automatic pilot for drones, specifically the incorporation of a system for detection and evasion of obstacles. This project is developed for the PX4 platform that is part of a project called DroneCode. DroneCode incorporates a series of hardware and software projects that work together for the flight of drones with an automatic pilot.

The main objective of this project is the treatment of events and analysis of the PX4 architecture. It has been decided to add a simple evasion system so that the main objective of this project has a usefulness and is the beginning of a project where this evasion system is improved to achieve a much more effective automatic pilot.

Although the main project focuses on PX4, this project will also address complementary technologies such as Mavlink or Gazebo, tools integrated in the DroneCode environment. Gazebo is the simulator that will be used in this project.

Keywords

PX4, drone, sensor, architecture, obstacle evasion.

Quiero dar las gracias

a mis padres por el apoyo económico y psicológico durante esta importante fase,

a mis compañeros Sergio, Alejandro, David José, Soulaimane que sin ellos no habría sido lo mismo esta etapa de mi vida,

a mi tutor por ayudarme en todo lo necesario dentro de este proyecto.

TABLA DE CONTENIDO

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

En este capítulo se presentará el proyecto. Primero se describirá la motivación que ha llevado a la realización del proyecto, posteriormente se describirán los objetivos del mismo y por último se explicará la estructura del presente documento.

1.1. Motivación

En la actualidad se está dando un gran auge de los sistemas de pilotaje automático, aunque aún no existen sistemas completos de piloto automático, ya que dependen de un piloto con los mandos en la mano para que un vehículo pueda ir en modo de piloto automático. Aviones y coches son vehículos que ya incorporan dichos sistemas como, por ejemplo, el sistema Autopilot de Tesla.

La utilización de pilotos automáticos en vehículos es una gran revolución. Desde el punto de vista de la seguridad vial, el uso efectivo de los pilotos automáticos ahorraría un gran número de víctimas e infraestructuras viales, ya que, principalmente en carretera, la gran mayoría de accidentes se dan por un factor humano. Aplicado al dron, el piloto automático amplificaría las posibilidades de éste como herramienta para acceder a lugares de difícil acceso para otros vehículos, siendo su manejo más fácil, rápido y accesible a un grupo más amplio ya que su pilotaje es sencillo de aprender y manejar.

Los trabajos para los que más se utilizan drones en la actualidad, son trabajos de grabación, particular o profesional, y para trabajos de seguridad, éstos últimos orientados al sector profesional. Un ejemplo de trabajos de seguridad para drones es la vigilancia del tráfico. Si se desarrollan pilotos automáticos para este tipo de trabajos, se podrán realizar de una forma más rápida y efectiva, sin la necesidad de pilotos que continuamente vigilen el dron. El dron podría controlar el tráfico de forma continua y ser equipado con un sistema de detección de accidentes o de control de velocidad máxima.

Empresas de transporte como Amazon también están investigando sobre el reparto a domicilio a través de drones con un piloto automático, en mi opinión este sistema es mucho más difícil de llevar acabo en un futuro cercano ya que la ley actual no permite estos vuelos sin piloto.

Este proyecto se centra en mejorar un sistema de piloto automático que actualmente está en desarrollo y en el que ya se pueden ejecutar misiones de forma autónoma, centrándose este proyecto en un sistema de evasión de obstáculos con un sensor de distancia tipo láser. Este tipo de pilotos automáticos en el futuro serán una herramienta que incorporen la mayoría de los vehículos por lo que es de gran interés investigar con ellos y desarrollar nuevos componentes.

1.2. Objetivo

Este proyecto se va a centrar en desarrollar un sistema de evasión de obstáculos que mejorará el sistema de piloto automático que ya cuenta con un funcionamiento lo suficientemente correcto, añadiendo para ello un sistema de detección y evasión de obstáculos. Aunque el objetivo es desarrollar un sistema de evasión, el proyecto se centrará en analizar la arquitectura y diseñar tareas secundarias que puedan trabajar junto al piloto automático. [1]

En concreto este trabajo se centrará en DroneCode, siendo éste un conjunto de proyectos que contienen todas las herramientas necesarias para el pilotaje autónomo de vehículos aéreos no tripulados. Dentro de este proyecto se modificarán dos de sus planteamientos y se utilizarán sin modificar el resto de ellos. PX4 y Gazebo serán los dos proyectos de DronCode a modificar, siendo PX4 el controlador de vuelo que maneja todas las acciones que debe realizar el dron para el vuelo autónomo. Este proyecto es una de las partes más importante de DroneCode ya que contiene el sistema de piloto automático y el sistema de control del dron. Por otro lado, Gazebo es un simulador especializado en robótica que permitirá generar escenarios y vehículos con los sensores y actuadores necesarios para realizar pruebas mediante software antes de utilizar el dron en pruebas reales.

1.3. Estructura

Este documento se divide en ocho capítulos que abordan la situación del piloto automático y el dron en el contexto actual, sus aplicaciones y posibilidades.

- **Introducción.** Este capítulo se centra en situar en contexto tanto el desarrollo de los pilotos automáticos en general como su aplicación al dron en particular, pasando por las variadas e interesantes aplicaciones y posibilidades del mismo.

- **Estado del arte.** Se realizará un estudio de las tecnologías y áreas de estudio de este proyecto. Este estudio permite analizar las herramientas que serán utilizadas en el proyecto y de esta forma agilizar el desarrollo y en algún caso elegir la herramienta más adecuada.
- **Diseño del sistema de evasión.** Este proyecto está basado en software por lo que se necesita un sensor de distancia de tipo laser diseñado en este apartado y ejecutado a través de Gazebo y un sistema de evasión creado dentro de PX4, el cual es el objetivo principal del proyecto.
- **Pruebas.** Se exponen un conjunto de supuestos para analizar y verificar el funcionamiento del sensor de distancia y del sistema de evasión.
- **Marco legal.** Se analiza la legislación española para el uso de vehículos aéreos no tripulados y aspectos legales en licencias de programas.
- **Entorno socioeconómico.** Se detallan tanto la planificación del proyecto como todas las estimaciones y recursos tanto económicos como materiales y humanos necesarios para llevarlo a cabo. También se justificará la realización del proyecto y los beneficios estimados que puede reportar la realización del mismo.
- **Conclusiones y trabajos futuros.** Después de la realización del proyecto se analizará las conclusiones del mismo, también se analizarán trabajos futuros, los cuales son muy importantes ya que este proyecto es una introducción al sistema PX4 donde hay muchos componentes que se podrán desarrollar.
- **Referencias.** En este capítulo se detallan las fuentes bibliográficas sobre las que se ha basado toda la información obtenida.

2. ESTADO DEL ARTE

En este capítulo se analizará e introducirán los conceptos básicos y la situación actual de cada una de las tecnologías que se usarán en el proyecto. Se hablará de los vehículos aéreos no tripulados y de los componentes necesarios para la realización del proyecto.

2.1. Vehículos aéreos no tripulados

Los vehículos aéreos no tripulados (UAV) o comúnmente conocidos como drones, son aeronaves que no necesitan ser tripuladas para su funcionamiento.

Los primeros drones surgieron en la primera guerra mundial como blanco en entrenamientos y como defensa contra zeppelins, aunque eran vehículos que no contenían todas las opciones de conducción de un vehículo común. Hasta finales del siglo XX no consiguen realizar vehículos con autonomía total [2].

Los drones actualmente están muy avanzados y se utilizan para una gran diversidad de aplicaciones que van desde el uso lúdico hasta investigaciones científicas, rescates, vigilancia fronteriza, uso en grabaciones comerciales, etc. Los usos más comunes son:

- En eventos: como partidos de futbol permiten realizar tomas aéreas desde ángulos que no pueden acceder las cámaras comunes.
- En situaciones de emergencia: ya que los drones tienen una gran capacidad de maniobra permite que éstos accedan a lugares donde el acceso humano o con otro tipo de vehículos es muy difícil o peligrosa.
- Zonas rurales: para agricultores estos dispositivos tienen un gran potencial ya que permite acceder a zonas de cultivo donde el acceso con otro vehículo podría dañar el cultivo y también puede recorrer varias hectáreas de una forma rápida tomando fotos o videos de alta calidad para control de plagas o encontrar posibles problemas del cultivo.
- Investigaciones biológicas: estas investigaciones se centran en las aves ya que el dron puede reproducir las rutas migratorias y permiten a los investigadores recoger información de estas rutas para realizar sus estudios.

- Recreativo: este uso cada vez es más común ya que los drones con el tiempo son cada vez más accesibles, para los aficionados a la filmografía o fotografía los drones son unas herramientas imprescindibles actualmente. También hay una modalidad que actualmente se está poniendo de moda que son las carreras de drones o los drones acrobáticos, estos drones son normalmente pequeños y muy rápidos que son controlados a través de un FPV.

La creciente oferta y comercialización de estas aeronaves está haciendo que cada vez aparezcan modelos más sencillos de manejar y de construcción más barata, por lo que se ha vuelto muy accesible a la población general, usándose de forma personal como vehículos de grabación, para realizar tomas aéreas.

Actualmente se están empezando a desarrollar drones que incorporan sistemas de piloto automático, aunque todavía no es un sistema muy desarrollado, Los sistemas actuales requieren de interacción humana o de otro sistema para marcar la ruta que el dron tiene que seguir, recayendo la resolución de problemas durante el vuelo también en la interacción humana. Toda esta interacción se realiza a través de estaciones de control. Como estos sistemas todavía se están desarrollando, no hay muchas aplicaciones, utilizándose por el momento para tareas simples como, por ejemplo, para vigilancia o tratamiento de tierras agrícolas.

Los pilotos automáticos son cada vez más comunes, y empresas dedicadas a la fabricación y desarrollo de drones como DJI, tienen sistemas de piloto automático propio, aunque también existen proyectos de código abierto que tienen sistemas de piloto automático muy avanzados como, por ejemplo, droncode o ardupilot. Estos pilotos automáticos todavía no tienen utilidades importantes ya que su desarrollo no es completo y la ley todavía no permite el vuelo de estos vehículos completamente autónomos.

2.2. C++

C++ es un lenguaje de programación creado en 1979, siendo este lenguaje una variante del lenguaje C pero con la particularidad que permite el manejo de objetos. C++ es un lenguaje multiparadigma ya que permite programación estructurada, programación orientada a objetos y programación genérica. Este proyecto está desarrollado en su mayor parte en C++, ya que el sistema estaba previamente desarrollado en este lenguaje. [3]

2.3. DroneCode

DroneCode es una solución software, hardware y de control para vehículos aéreos no tripulados (UAV). Este sistema contiene un sistema de piloto automático para el UAV, que funciona conjuntamente con una estación de control y un ordenador que controla todo el hardware disponible del dron. El objetivo de DroneCode es ser una plataforma mejor, más segura, fácil de usar y flexible que otros proyectos de código abierto o privativos. DroneCode trabaja junto a su propia comunidad y a la Fundación Linux para expandirse.

DroneCode está formado por un conjunto de proyectos de código abierto o hardware abierto, estos proyectos son:

- PX4: control de vuelo de DroneCode.
- QGroundControl: estación de control de DroneCode.
- MAVLink: sistema de comunicación entre la estación de control y el vehículo.
- Pixhawk: hardware donde se ejecutará el control de vuelo, también está adaptado a Qualcomm Snapdragon Flight e Intel Aero Ready.
- Gazebo: simulador software, también disponible JMavSim y AirSim.

2.4. Mavlink

Mavlink es un protocolo de comunicación basado en una librería ligera que interacciona con la estación de control y el sistema de vuelo PX4 [4]. También es utilizado para la comunicación entre PX4 y los distintos simuladores. Mavlink está basada en estructuras C y tiene una gran cantidad de mensajes preconfigurados, pero también se pueden crear nuevos mensajes personalizados.

Los mensajes que utiliza Mavlink son creados a partir de ficheros .xml que permiten reutilizar las estructuras en desarrollos con diferentes soluciones software para drones. En el proyecto actual se utilizará Mavlink para la comunicación entre la simulación y el sistema de control de vuelo, para ello se utiliza un mensaje preconfigurado en Mavlink que envía la información de un sensor de distancia.

2.5. QGroundControl

QGroundControl es una estación de control para PX4 y Ardupilot, que permite planificar una misión y control completo para cualquier dron conectado a través de Mavlink. También permite un uso sencillo para empezar a usarlo y posteriormente un uso profesional.

QGroundControl es un sistema multiplataforma disponible para la mayoría de los sistemas: Linux, Windows, MacOs, Android y IOS. En el este proyecto se ha utilizado la versión de QGroundControl para Linux y para MacOs, las cuales no tienen apenas diferencias y serán utilizadas para programar la misión que deberá realizar el dron y también se utilizará para guardar el progreso de la misión.

2.6. Pixhawk

Pixhawk es un proyecto de hardware abierto que permite la ejecución de un sistema de piloto automático a través de PX4 y Ardupilot y que, con un coste bajo, obtiene altas prestaciones y es fácil de utilizar [5]. Pixhawk ha creado varias versiones, todas basadas en diseños públicos con la intención de optimizar el funcionamiento y de que se pueda usar en carreras en primera persona (FPV), para tareas de procesamiento intenso es recomendable separar ese procesamiento en otro computador como por ejemplo una Raspberry pi. En el proyecto que vamos a realizar, el procesamiento llevado a cabo no es excesivo por lo que no necesitamos un procesador externo.

La versión que se utilizará en este proyecto es la Pixhawk 2.4.8, versión de 32 bits, que cuenta con una memoria de 256 KBytes, un giroscopio, dos acelerómetros y un barómetro. No es la última versión de Pixhawk, pero es válida para la realización del proyecto.

2.7. Simulador

Hay dos tipos de simuladores: Software In the Loop (SITL), donde la simulación se realiza únicamente por un ordenador o varios en la misma red, y Hardware In the Loop (HITL) en el cual la simulación se realiza en la placa controladora de vuelo.

En este proyecto se utilizarán pruebas Software In the Loop (STIL), por lo que se profundizará en este tipo de pruebas. A continuación, se mostrará un esquema de la arquitectura del simulador.

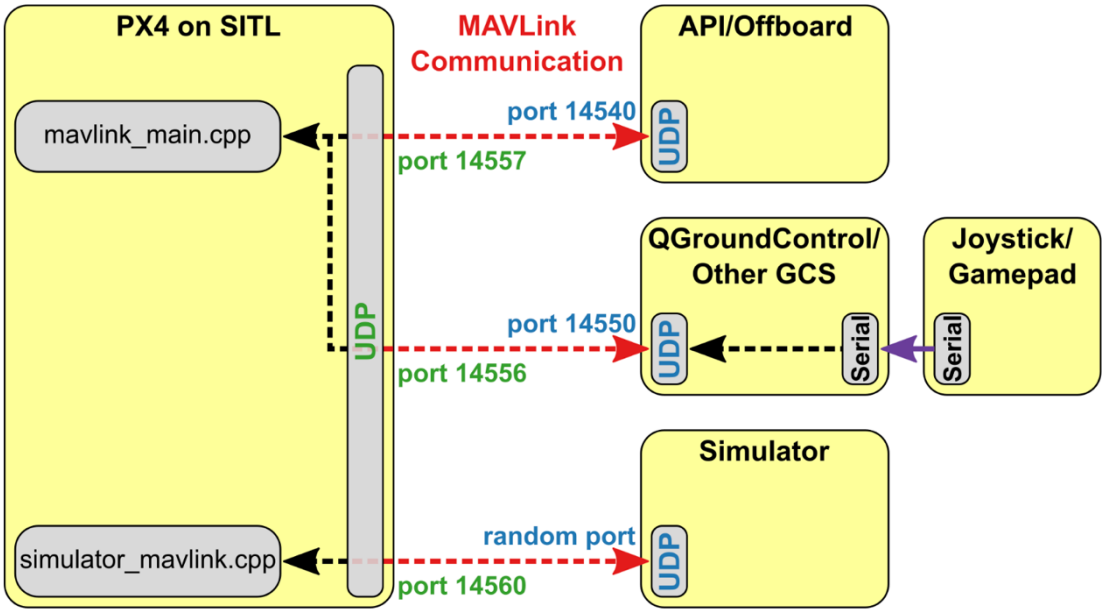


Imagen 1 Estructura simulador/PX4 [6]

Como podemos ver en la *Imagen 1*, la comunicación entre el simulador y PX4 se realiza a través de Mavlink, concretamente a través del puerto 14560 con protocolo UDP, El simulador envía la información de los sensores a un módulo de PX4 dedicado a la información del simulador, y este módulo responde con la información de los actuadores para que el simulador pueda representar la actuación del dron.

En DroneCode existen varios simuladores, en este proyecto se han probado dos de ellos, JMavSim y Gazebo, que a continuación, se analizarán y se decidirá cuál utilizar en este proyecto.

2.7.1. JMavSim

JMavSim es un simulador basado en java. [7] Es el simulador inicial con el que se empezó este proyecto ya que tiene una fácil instalación y el rendimiento es óptimo, pero tras una amplia documentación sobre este simulador se vio que apenas tenía información de funcionamiento y desarrollo, por lo que al intentar aplicarlo al sensor de distancia, la falta de información hizo que este simulador quedara fuera del proyecto.

2.7.2. Gazebo

Gazebo es un simulador de robótica fácil de usar que permite mediante un conjunto de herramientas realizar un diseño de una forma rápida y sencilla y con este diseño realizar una serie de pruebas para saber si los algoritmos funcionan correctamente. Gazebo incluye una simulación tanto de ambientes exteriores como en interiores. También contiene un potente motor físico que permite realizar una simulación precisa. Gazebo además incluye un conjunto de tutoriales que están disponibles en su web para empezar a utilizarlo de una forma sencilla. A parte de todo esto, Gazebo es gratuito y dispone de una gran comunidad que crea muchos modelos que posteriormente podemos utilizarlos. [8]

Al contrario que JMavSim, Gazebo tiene una gran comunidad de desarrollo y una página donde existen una serie de tutoriales y documentación para el manejo y desarrollo de nuevos componentes, después de realizar estos tutoriales se empezará a crear el sensor.

En el proyecto que se está realizando se va a usar como simulador Gazebo, ya que es el simulador más completo y que tiene una mayor cantidad de escenarios y de vehículos de los simuladores utilizados en DroneCode, en posteriores apartados se explicará detalladamente que se ha utilizado para realizar este proyecto.

2.8. PX4

PX4 es un proyecto de código abierto, el piloto automático está realizado íntegramente en el lenguaje de programación C++. Este proyecto contiene un sistema de guía, navegación y control de los algoritmos para el sistema autónomo de vuelo, siendo válido para vehículos no tripulados multirotor y aeroplaneadores.

PX4 es un sistema creado para diferentes proyectos y aunque todavía la industria los vehículos aéreos se encuentra en pleno desarrollo, ya hay varios casos de uso para estos sistemas, tanto para consumidores, como para uso industrial. Por ejemplo, PX4 tiene una opción de seguir a un usuario a través del teléfono móvil, este sistema se puede usar para realizar videos sin tener que pilotar el vehículo. También se plantean usos de transporte de cargamentos, para vigilancia de edificios o de cultivos y otros muchos usos que están por llegar.

PX4 es el sistema donde se alojará la mayor parte de este proyecto ya que es el sistema donde se encuentra el piloto automático que controla al dron, y en este piloto automático es donde se integrará el sistema de tratamiento de eventos. Para ello se ha investigado cómo funciona el sistema PX4. A continuación, se detallará como es el sistema de PX4 en profundidad.

PX4 contiene dos capas principales, un estimador o simulador de vuelo y un controlador de vuelo. Todos los vehículos compatibles con PX4 contienen el mismo código base lo que permite tener varios vehículos con una configuración muy sencilla. La comunicación entre todos los componentes del sistema se realiza a través de un sistema de comunicación genérico llamado uORB, sistema de comunicación del que hablaremos posteriormente ya que el proyecto lo utiliza. En el siguiente la *Imagen 2* podemos ver un esquema con el funcionamiento de PX4, donde toda la parte superior del sistema constituye el middleware, digamos que sería la capa que mantiene a todos los componentes unidos. La parte inferior del esquema corresponde al Flight Stack. En la *Imagen 3* se verá de forma más detallada.

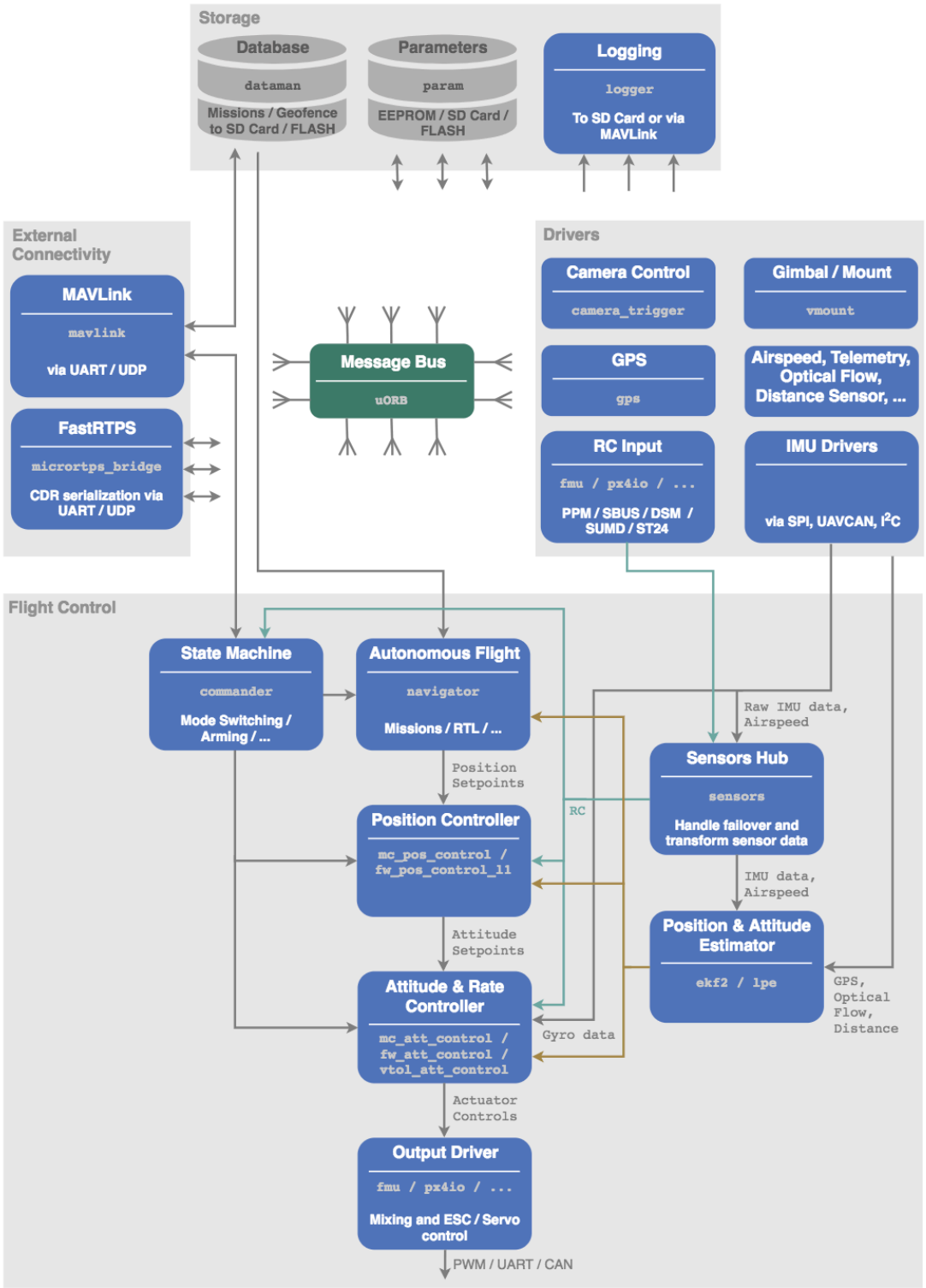


Imagen 2 Estructura de PX4 [9]

2.8.1. Flight Stack

El flight stack es un conjunto de algoritmos de estimación, control y de actuación para el control autónomo de un dron. A continuación, se detallarán cada una de las partes:

- Estimador: recoge la información de uno o diferentes sensores para generar un estado del vehículo, por ejemplo, a partir de diferentes sensores como el GPS, el barómetro, etc. Este estado sería la altura del vehículo.
- Controlador: el objetivo de este controlador es estimar una posición real a partir de una posición simulada y un estado estimado en el estimador anteriormente descrito, generando a partir de esos datos una nueva posición corregida.
- Actuador: este algoritmo toma los comandos de movimiento y transforma estos comandos en la actuación física que deben realizar los motores para cumplir con la misión del comando, también controla que no exceda los límites permitidos por los motores.

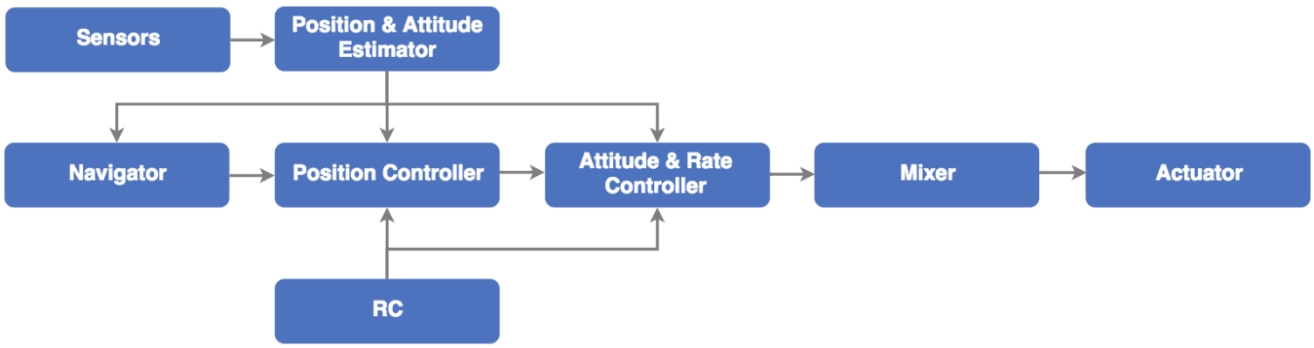


Imagen 3 Diagrama de componentes [9]

En la *Imagen 3*, se muestra un diagrama de cómo se integran todos los componentes anteriormente descritos. Un ejemplo de actuación de este sistema sería el siguiente:

A partir de la información del barómetro, se puede obtener la altura del dron, esta altura ha cambiado y el controlador decide que el dron tiene que aumentar la altura y envía al actuador la acción de elevar el dron, por último, el actuador se encarga de aumentar la velocidad de los motores para aumentar la altura.

2.8.2. Middleware

El middleware consta principalmente de los drivers que permiten a los sensores transformar la información física a información digital, también se encarga de enviar los datos de los drivers a los diferentes componentes de PX4, utilizando para esta comunicación uORB, el cual es un sistema de suscripción/publicación que permite a los componentes comunicarse entre sí de una forma sencilla.

La velocidad de actualización depende del módulo que lo requiera, los drivers tienen una velocidad de actualización mayor, normalmente de 1 kHz, aunque normalmente utilizan una velocidad de 250 Hz. También existen otras partes del sistema como el navegador que no necesitan una velocidad de actualización tan rápida.

El middleware también incluye el modulo que se comunica con el simulador y adapta la información facilitada por el mismo de tal forma que el sistema contenga la información necesaria para funcionar como si estuvieran todos los componentes conectados.

2.8.3. Entorno de ejecución

PX4 se ejecuta en varios sistemas operativos, todos basados en Posix como, por ejemplo, Linux, MacOs o NuttX. Con una política basada en un sistema de tiempo real. Hay dos formas principales de ejecución que utilizan los módulos del sistema.

- Tarea: el módulo ejecuta su propia tarea con su propia pila y prioridad.
- Cola de trabajo: el módulo se ejecuta de forma que comparte tareas, que no tiene su propia pila, sino que comparten la misma pila de memoria y la prioridad depende de la cola de trabajo a la que pertenezcan cada una de las tareas.

La ventaja de utilizar tareas es que consumen menos memoria RAM ya que se ejecutan en el periodo de tiempo que se ha seleccionado previamente, de lo contrario, las tareas no permiten quedarse suspendidas hasta la llegada de un mensaje.

Las colas de trabajo se usan para tareas periódicas como, por ejemplo, la lectura de mensajes procedentes del mando de control como el detector de aterrizaje.

PX4 permite el uso de tareas en segundo plano, esto es muy importante para el proyecto ya que nos va a permitir programar una nueva tarea, que se ejecute en segundo plano leyendo la información del sensor y actuando en casos de peligro para la integridad del vuelo. En el caso de estar ejecutando dentro del sistema operativo NuttX se ejecutaría una nueva tarea, y en el caso de ejecutar en Linux se ejecutaría un nuevo hilo de ejecución. En este hilo o tarea se puede especificar los parámetros de funcionamiento que se necesita: el tipo de planificador de tareas, la prioridad de la tarea y el tamaño de la pila de memoria.

2.8.4. NuttX

NuttX es el sistema operativo donde se ejecuta PX4 cuando es ejecutado a bordo del control de vuelo, en este caso, Pixhawk. NuttX es un sistema operativo de tiempo real, muy estable, que requiere poco espacio en la memoria y es eficiente. Es un sistema escalable de 8 a 32 bits dependiendo el entorno de ejecución que se requiera. Como ya sabemos, está basado en POSIX, pero también cumple los estándares de ANSI. La principal desventaja de NuttX es que el resto de APIs disponibles en Linux o comunes en sistemas operativos de tiempo real no están disponibles en NuttX o no funcionan correctamente.

NuttX ejecuta cada módulo de PX4 como una tarea independiente, cada tarea puede ejecutar uno o varios hilos lo que permite una mayor libertad a la hora de diseñar nuevos módulos como, por ejemplo, el que se usaría para el proyecto actual. Cada tarea tiene un espacio fijo de memoria y un periodo fijo de ejecución, para mejorar la eficiencia del procesado se comprueba si hay suficiente espacio en la pila de ejecución y en el caso de que se necesite se usa “Stack Coloring”, que es una técnica que permite usar el mismo espacio de ejecución virtual para distintas tareas.

La principal diferencia entre NuttX y Linux es que, en Linux, PX4 se ejecuta en un único proceso y cada módulo se ejecuta con sus propios hilos, sin diferenciar entre tarea e hilos como hace NuttX.

2.8.5. Módulos de comunicación

En PX4 hay dos modos de comunicación principales, Mavlink y uORB, uORB es el encargado de la comunicación interna de PX4, dentro del mismo espacio de ejecución, mientras que Mavlink se encarga de comunicaciones externas entre PX4 y otros sistemas como, por ejemplo, QGroundControl. En el proyecto actual, como la ejecución del mismo se realiza como tarea secundaria en PX4, se utilizará uORB ya que está dentro del sistema de PX4. En el caso de haber necesitado más recursos para la ejecución del módulo, se podría haber ejecutado en un hardware externo y enviar las acciones, en este caso comandos, a través de Mavlink. A continuación, se explicará uORB, ya que es un sistema que únicamente trabaja dentro de PX4, Mavlink se ha explicado anteriormente ya que está integrado en otros proyectos a parte de en PX4.

2.8.5.1. uORB

uORB es un sistema de comunicación asíncrono basado en la publicación y subscripción de mensajes para la comunicación entre procesos o hilos. uORB permite la creación de nuevos mensajes de una forma sencilla, pero en el caso del proyecto actual no se necesitará añadir nuevos mensajes, ya que se usarán los mensajes ya predefinidos dentro de PX4 [10].

Para este proyecto se necesita la subscripción a un conjunto de mensajes necesarios para el tratamiento de la información recibida por el sensor, y otros parámetros como, por ejemplo, el modo de funcionamiento del dron o la posición actual del dron.

Para el proyecto solo se usará la publicación de comandos donde se indicará que acción necesita realizar el dron para la evasión de los obstáculos. En capítulos posteriores se explicará detalladamente qué datos se recogen y para qué se utilizan.

La documentación de PX4 incluye un gráfico donde podemos ver todos los mensajes que están incluidos en uORB y qué módulos usan estos mensajes.

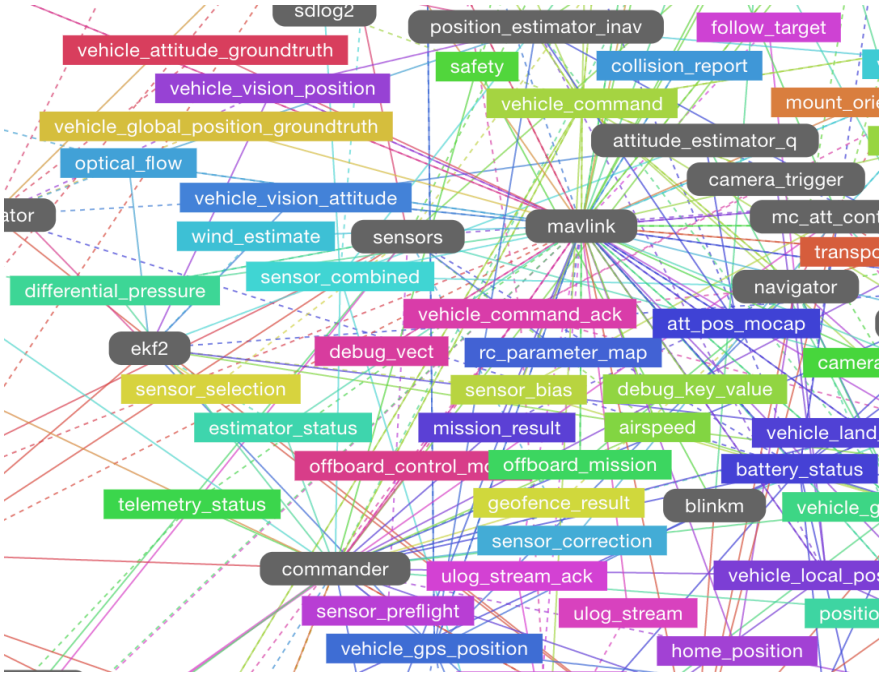


Imagen 4 Ejemplo de uORB [11]

La *Imagen 4* muestra el gráfico de uORB, que aunque parezca difuso, tiene un filtro que ayuda a poder manejarlo fácilmente. Como se puede ver, los mensajes de uORB se muestran de diferentes colores, mientras que los módulos de PX4 son de color gris.

2.8.6. Funcionalidad

Después de describir todas las partes del sistema, vamos a analizar la funcionalidad actual que tiene PX4. Toda la configuración del plan de vuelo y del vehículo se puede realizar a través de QGroundControl de una forma visual. PX4 permite realizar vuelos de forma automática, aunque hay algunos inconvenientes como, por ejemplo, que el despegue y aterrizaje de la aeronave solo está disponible para aeronaves de alas fijas, como la que podemos ver en la *Imagen 5*. En el resto de los vehículos es muy recomendable hacer el despegue de forma manual y una vez está el vehículo en el aire empezar la misión.

Imagen 5 Aeronave de ala fija

Para realizar cualquier tipo de acción manual, PX4 permite la configuración de un mando por radiofrecuencia y de tres modos de vuelo manual que facilitan las acciones manuales [12]:

- Estabilizador: este modo es principalmente para usuarios de nivel bajo, ya que facilita el control del vuelo evitando giros bruscos y manteniendo el dron en vuelo si los sticks del mando se sueltan.
- Altitud: este modo controla que la aeronave no ascienda o descienda a una velocidad mayor a la indicada en el control del modo. Por ejemplo: si se configura una velocidad de ascensión de 0.3 metros por segundo, nunca se podrá superar esa velocidad, aunque le indiquemos mediante el mando la máxima potencia.

- Posición, este modo es muy útil ya que si los sticks del mando se sueltan el dron mantendrá su posición en el aire sin mover ningún stick. La posición se mantiene incluso si el aire mueve el dron, ya que el dispositivo corrige la posición mediante GPS.
- Acrobático, este modo permite mediante los sticks del mando controlar el ángulo del dron pudiendo realizar acrobacias, cuando los sticks están centrados, el dron mantiene el ángulo en el que se encuentra.
- Acrobático/estabilizador, este modo, que lo llaman "Ratitude" es una mezcla entre el modo acrobático y el estabilizador, Cuando los sticks se encuentran en la circunferencia interior del mando, aproximadamente un 80% de la longitud de los mandos, el dron se comporta en modo de estabilización, pero si se supera esa circunferencia interior, el dron entra en modo acrobático.
- Pausa, este modo mantiene la posición del dron en el aire, es muy parecido al modo de posición, pero en este modo si tocamos los botones de movimiento del mando de control, el dron no realizará ninguna acción. Este modo se utilizará en el proyecto, ya que si en los parámetros del comando, se especifica una posición GPS, el dron irá a esa posición de forma automática y se mantendrá una vez llega a su destino.
- Despegue, PX4 tiene un modo de despegue automático donde se puede configurar la altura y la velocidad de ascensión. Si el dron está armado, automáticamente asciende hasta dicha posición y se mantiene a la espera de nuevas órdenes a través del mando de control.
- Aterrizaje, al igual que en el despegue, se puede configurar la velocidad de descenso y el tiempo que permanece en el suelo hasta que el dron se desarma, Por defecto, el dron no se desarma una vez ha aterrizado, aunque se puede configurar.
- Regreso, este modo devuelve el dron a su posición de casa, que debe ser configurada previamente, si no es configurada, será la posición donde el dron es armado.

Aunque los modos manuales son una buena opción para empezar a manejar el dron, el modo más importante de PX4 son las misiones, pudiendo configurar una misión a través de QGroundControl, en cuyo caso el dron ejecutará la misión marcada sin ninguna acción humana durante la misma. Hay varios tipos de misiones que el dron puede realizar de forma automática, aunque en este proyecto nos vamos a centrar en un tipo de misión únicamente que será “rally point”.

Este tipo de misión consiste en marcar un conjunto de puntos en el mapa que pueden ser configurados con distintos parámetros como, altura, velocidad o mantener la posición un tiempo determinado al llegar al punto de control. Una vez la misión es configurada en QGroundControl se envía al dron de una forma automática y cuando pulsemos el botón de iniciar misión, el dron iniciará el viaje al primer punto de control de forma automática. A continuación, la *Imagen 6* muestra una misión ya configurada.

Imagen 6 Ejemplo de misión

3. DISEÑO DEL SISTEMA DE EVASIÓN

En este capítulo se analizará el sistema y se planteará una propuesta de solución para la realización del sistema de evasión, posteriormente se detallará el proceso de creación de todos los componentes necesarios para la realización del proyecto.

3.1. Análisis del sistema

A continuación, se planteará la solución inicial del sistema, la cual se basa en un sistema de evasión en un dron simulado por parte de Gazebo. El proyecto se va a centrar en analizar la ejecución de una misión como se explicó en el apartado *2.8.6 Funcionalidad*.

PX4 permite ejecutar misiones que se basan en una serie de puntos geográficos por los que el dron debe pasar hasta completar la misión, la cual puede pararse en cualquier momento y mandar una acción distinta al dron por medio de QGroundControl o por el mando previamente configurado. En ninguno otro caso el dron detendrá la misión. Esto es un problema ya que la parte importante del sistema es que el dron pueda volar de forma autónoma, pero si un piloto tiene que estar vigilando el dron por posibles problemas, el dron no volaría de una forma autónoma efectiva.

Para mejorar este sistema de misión se ha decidido plantear un sistema de detección y evasión de obstáculos. Ya que es la primera vez que se trabaja con este sistema, la mayor parte del proyecto se ha dedicado al análisis de la arquitectura de PX4 y cómo interaccionar con ella para poder añadir soluciones como la planteada en este proyecto.

En concreto, para el sistema de detección y evasión se necesitará la creación de un sensor de distancia de tipo laser que detecte objetos con los que el dron pueda colisionar, También se necesita crear una tarea dentro del PX4 que analice los datos del sensor y transforme esos datos en órdenes sobre la misión para que el dron no sufra ningún accidente.

3.2. Requisitos del sistema

Para la realización del siguiente proyecto se necesitan definir un conjunto de requisitos, funcionales y no funcionales. Estos requisitos se dividirán en dos partes, la parte del simulador, y la parte del control de vuelo, aunque ambas partes forman parte del mismo proyecto.

Los requisitos funcionales son aquellos que especifican las capacidades o funciones que debe realizar el sistema, mientras que los requisitos no funcionales se encargan de especificar las restricciones o características funcionales del sistema.

Para la especificación de los requisitos se usará unas tablas como la *Tabla 1*, donde podemos ver un ejemplo. La tabla se compone de los siguientes campos:

- Identificador: cada requisito contiene un identificador que facilita la trazabilidad del mismo. Este código tendrá el formato RSX-Y-ZZ, donde RS indica que es un requisito del sistema, la X hace referencia a si el requisito es funcional o no funcional en cuyo caso tomará el valor de F o NF respectivamente. La Y hace referencia a los dos grandes bloques del proyecto, S para el simulador, PX4 para el controlador de vuelo y, por último, ZZ indica el número de requisito.
- Título, breve descripción del requisito.
- Descripción, texto explicativo sobre la funcionalidad o restricción del requisito.
- Prioridad, se establece un sistema de prioridad por requisito para posteriores tomas de decisiones según la importancia de los mismos. También facilita el orden para los programadores dividiendo la prioridad en alto, medio y bajo.
- Necesidad, en este apartado se especificará la importancia del requisito, utilizando para una mejor organización en el desarrollo del sistema tres niveles de importancia: obligatorio, necesario y opcional.
- Fuente, se indicará el origen del requisito.

RSX-Y-ZZ				
Título				
Descripción				
Prioridad			**Necesidad**	
☐Alta	☐ Media	☐ Baja	☐ Obligatorio	☐ Necesario ☐ Opcional
Fuente				

TABLA 1 PLANTILLA DE REQUISITOS DEL SISTEMA

RSF-S-01				
Título	Integrar sensor			
Descripción	El simulador debe integrar un sensor de distancia.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 2 RSF-S-01

RSF-S-02				
Título	Sensor de tipo laser.			
Descripción	El sensor que implementa el simulador debe ser un sensor de distancia basado en láser.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesari ☐ Opcional
Fuente	Tutor			

TABLA 3 RSF-S-02

RSF-S-03				
Título	Sensor rotatorio			
Descripción	El sensor debe ser rotatorio para tener visualización en los 360º alrededor del dron.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 4 RSF-S-03

RSF-S-04				
Título	Integración con el dron			
Descripción	El sensor debe estar integrado en el cuerpo del dron.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 5 RSF-S-01

RSF-S-05				
Título	Distancia y orientación			
Descripción	El sensor debe ser capaz de obtener la distancia y orientación de objetos que se encuentren en su campo visual.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 6 RSF-S-05

RSF-S-06				
Título	No detección			
Descripción	Si el sensor de distancia no detecta ningún objeto en su campo de visión, mostrará la distancia máxima soportada por el mismo.			
Prioridad			**Necesidad**	
☐Alta	☑ Media	☐ Baja	☐ Obligatorio	☑ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 7 RSF-S-06

RSF-S-07				
Título	Enviar información			
Descripción	El sensor debe enviar la información recogida al controlador de vuelo.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 8 RSF-S-07

RSNF-S-01				
Título	Sistema de envío			
Descripción	El sensor utilizará Mavlink para enviar la información necesaria para el correcto funcionamiento del sistema.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 9 RSNF-S-01

RSNF-S-02				
Título	Datos de envío			
Descripción	Por restricciones del sistema, se debe enviar la información divida en 4 fragmentos de 90 grados cada uno.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 10 RSNF-S-02

RSNF-S-03				
Título	Rendimiento correcto			
Descripción	Una vez el sensor está integrado en el sistema, el mismo debe ejecutarse de una forma totalmente funcional y con una buena experiencia para el usuario.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 11 RSNF-S-03

RSNF-S-04				
Título	Integración con el sistema			
Descripción	El sensor debe estar integrado totalmente con el sistema, para que el mismo funcione de la misma forma que funcionaba antes de integrar el sensor de distancia.			
Prioridad			**Necesidad**	
☐Alta	☐ Media	☑ Baja	☐ Obligatorio	☑ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 12 RSNF-S-04

RSF-PX4-01				
Título	Comprobar fase			
Descripción	Se debe comprobar la fase en la que se encuentra el dron, si no está en la fase de misión automática, el dron no debe ejecutar ninguna acción			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 13 RSF-PX4-01

RSF-PX4-02				
Título	Comprobar datos del sensor			
Descripción	El sistema debe comprobar los datos del sensor, si existen deberá almacenarlos y empezar con el protocolo de protección.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 14 RSF-PX4-02

RSF-PX4-03				
Título	Analizar velocidad			
Descripción	Calcular la velocidad de desplazamiento del dron, para usarlo posteriormente en las maniobras de evasión.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 15 RSF-PX4-03

RSF-PX4-04				
Título	Distancia de seguridad			
Descripción	La distancia de seguridad que se usará para las fases de prevención del sistema de evasión se calculará en función de la velocidad.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 16 RSF-PX4-04

RSF-PX4-05				
Título	Obtener inclinación del vehículo			
Descripción	El sistema de poder conocer la inclinación del vehículo para posteriormente calcular la distancia con el suelo.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 17 RSF-PX4-05

RSF-PX4-06				
Título	Calcular altura			
Descripción	El sistema debe poder calcular la diferencia de altura entre el objeto detectado y el dron. La altura se calculará con la siguiente fórmula: $\sin(p) * D$, donde p es el ángulo de inclinación del dron y D la distancia obtenida por el sensor de distancia.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 18 RSF-PX4-06

RSF-PX4-07				
Título	Fases del sistema.			
Descripción	El sistema debe contar de tres fases: no peligro, peligro y evasión.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 19 RSF-PX4-07

RSF-PX4-08				
Título	Sin peligro			
Descripción	Esta fase es la primera del sistema de seguridad, en esta fase únicamente se comparará la distancia obtenida por el sensor de distancia y la distancia de seguridad, si la distancia del sensor es menor que la distancia de seguridad empezará la fase de peligro.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 20 RSF-PX4-08

RSF-PX4-09				
Título	Peligro			
Descripción	En esta segunda fase se comparará la distancia del sensor con la distancia de seguridad y también se comprobará que el objeto detectado no esté a la misma altura que el suelo. Si el objeto está por encima del suelo (no es falsa lectura), y la distancia del sensor es menor que la distancia de seguridad pasará a la fase de evasión.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 21 RSF-PX4-09

RSF-PX4-10				
Título	Error peligro			
Descripción	Si la lectura de la fase “No peligro” era falsa, en la fase de peligro se detectará, con tres lecturas falsas dentro de la fase de peligro, el sistema volverá a la fase de “No peligro”			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 22 RSF-PX4-10

RSF-PX4-11				
Título	Evasión			
Descripción	En esta fase el dron comenzará la maniobra de evasión, una vez el dron termine la maniobra de evasión, se volverá a la fase de “No peligro”			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 23 RSF-PX4-11

RSF-PX4-12				
Título	Maniobra de evasión			
Descripción	La evasión se realizará de forma perpendicular al objetivo.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 24 RSF-PX4-12

RSF-PX4-13				
Título	Sentido de la evasión			
Descripción	El sentido de la evasión dependerá de la información obtenida por el sensor de distancia en la dirección a esquivar.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 25 RSF-PX4-13

RSF-PX4-14				
Título	Altura de la evasión			
Descripción	Como el sensor de distancia solo detecta objetos en el mismo plano que se encuentra el dron, la evasión se realizará también a la misma altura a la que se encuentra el dron.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 26 RSF-PX4-14

RSNF-PX4-01				
Título	Rendimiento			
Descripción	El rendimiento del Pixhawk no debe ser afectado por el nuevo sistema de evasión			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 27 RSNF-PX4-01

RSNF-PX4-02				
Título	Errores controlados			
Descripción	Los errores producidos por el nuevo sistema de evasión deben estar controlados para no afectar al sistema general de PX4.			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 28 RSNF-PX4-02

RSNF-PX4-03				
Título	Memoria			
Descripción	La memoria utilizada en la nueva tarea donde se implementa el sistema de evasión no debe superar los 2 MB			
Prioridad			**Necesidad**	
☑Alta	☐ Media	☐ Baja	☑ Obligatorio	☐ Necesario ☐ Opcional
Fuente	Tutor			

TABLA 29 RSNF-PX4-03

3.3. Diseño de la solución

Para poder entender correctamente la solución propuesta para el sistema de evasión de obstáculos, vamos a colocar todos los módulos del proyecto en su correspondiente parte del sistema.

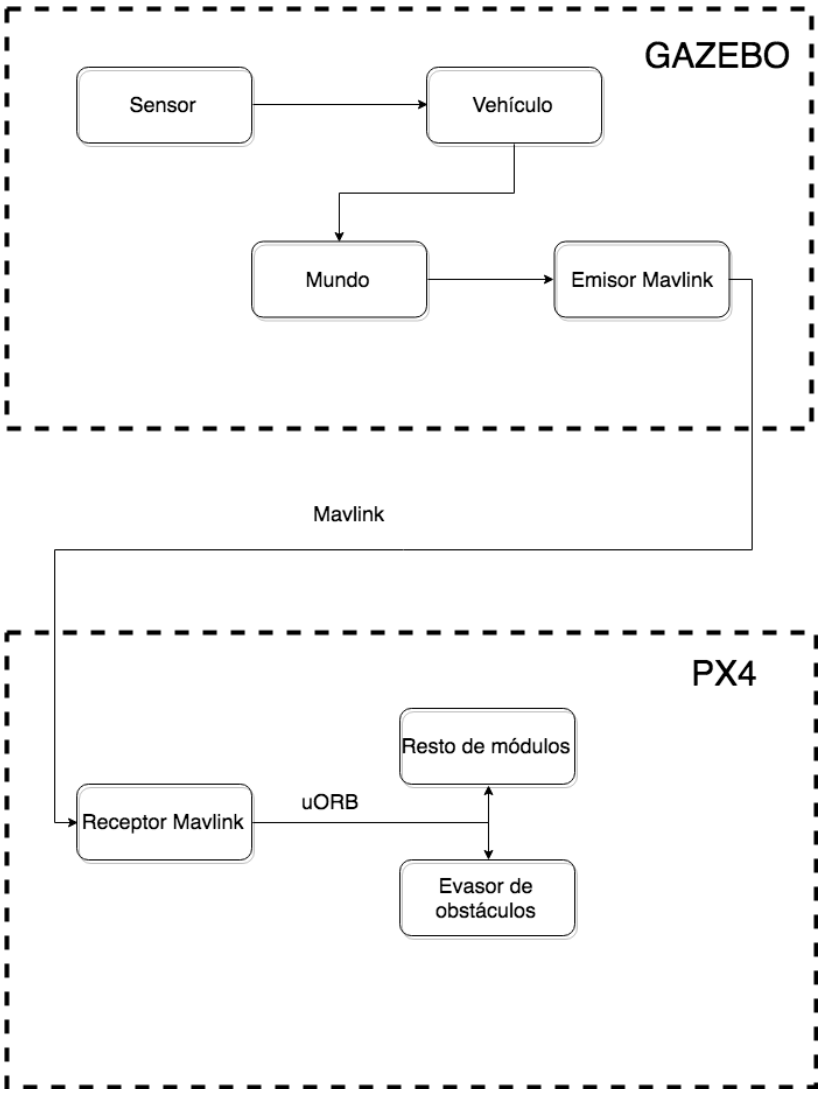


Imagen 7 Estructura del nuevo componente

Como podemos ver en la *Imagen 7*, dentro del simulador debemos añadir el módulo del sensor, el cual forma parte del vehículo, y a su vez, el vehículo forma parte de un mundo en el que tenemos todos los módulos del escenario de la simulación y otros módulos como el de comunicación a través de Mavlink. Al estar el módulo de comunicación y el módulo del sensor dentro del mismo mundo, se pueden comunicar entre ellos para enviar la información a PX4 través de Mavlink con el mensaje de sensor de distancia ya predefinido.

La recepción del mensaje en PX4 del sensor de distancia ya está implementada, por lo que debemos subscribirnos al mensaje de sensor de distancia de uORB para obtener los datos enviados desde el simulador. Para ejecutar las acciones de evasión también se utilizará uORB con el mensaje de comandos.

También hay una alternativa a esta solución para los casos en los que el sistema desarrollado necesite una capacidad de cómputo muy grande y el Pixhawk dispone de una capacidad de cómputo tan grande. [13] Para solucionar este problema se puede analizar los datos en otro computador, por ejemplo, una Raspberry Pi y comunicarse con PX4 a través de Mavlink.

Este cambio necesitaría modificar la comunicación que se hace en este proyecto, ya que para la comunicación interna se utiliza uORB para optimizar los recursos de Pixhawk pero en el caso de que separar el desarrollo en un dispositivo externo se debe cambiar a Mavlink, tanto los mensajes de lectura como los de escritura. Los comandos utilizados en este sistema son todos reproducibles con Mavlink por lo que no es necesaria una gran modificación del sistema para la implementación en un dispositivo externo.

A continuación, empezaremos explicando el desarrollo del sensor de distancia y cómo ha sido implementado, y posteriormente seguiremos con el módulo de evasión de obstáculos.

3.3.1. Sensor de distancia

Como hemos dicho en el apartado *2.7.2 Gazebo* se explica como es el simulador de DroneCode que más entornos y vehículos ofrece, para este proyecto se ha utilizado el siguiente modelo:

Imagen 8 Diseño del cuerpo del dron

El vehículo que podemos observar en la *Imagen 8*, es un cuadricóptero, es decir, un vehículo aéreo de la familia de los helicópteros con cuatro hélices, movidas por un motor en cada hélice. Estos motores están anclados cada uno a un brazo que emerge del cuerpo del vehículo, y las hélices realizan una fuerza de empuje hacía una dirección, normalmente hacía abajo, lo que permite el movimiento del vehículo.

Este modelo no implementa únicamente el modelo físico del dron, también implementa todos los sensores y motores necesarios para el modelado del sistema, como por ejemplo, el GPS, el barómetro necesario para predecir la altura a la que se encuentra el vehículo, etc.

Para la detección de obstáculos es necesario modificar el modelo anterior ya que no implementa ningún sensor que permita la detección del obstáculo. Para este fin se necesita un sensor de distancia, el cual puede ser de varios tipos:

- Láser: Este tipo de sensores detectan los objetos a través de una emisión de luz y se componen de un emisor y un receptor de luz, El emisor de luz proyecta un haz de luz (láser), que cuando es reflejado en un objeto rebota y es captado por el receptor de luz. A continuación, en la *Imagen 9*, podemos ver cómo funciona un sensor de distancia de tipo laser.

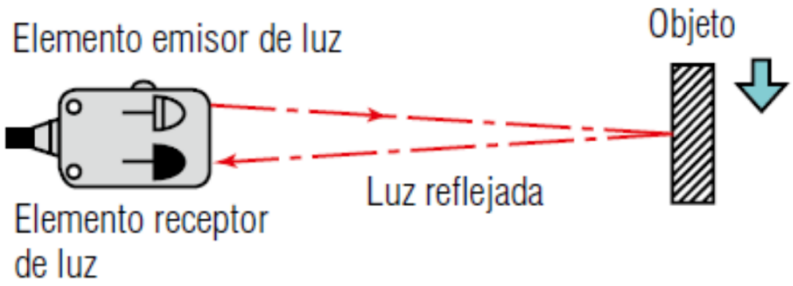


Imagen 9 Esquema sensor distancia

Si el receptor de luz no capta señal, significa que no hay ningún objeto en el campo de visión del sensor. Para la medición de la distancia se utilizan distintas técnicas, una de las más utilizadas es medir el tiempo que tarda en enviar el haz de luz y el receptor en captarlo, de esta forma se puede saber la distancia a la que se encuentra el objeto.

- Radar: El sensor de distancia RADAR funciona a través de ondas electromagnéticos, se utiliza para medir distancia, altitud, dirección, etc. El funcionamiento consiste en emitir un impulso magnético que es reflejado en un objeto y se recibe en la misma posición que fue emitido. A partir de este impulso se puede obtener información de la distancia y la posición del objeto. [14]

Para el proyecto se ha planteado la opción de usar un sensor láser de 360º ya que el radar necesita un cómputo mayor teniendo que utilizar otro sistema de cómputo para analizar la información obtenida por el sensor. El sensor laser de 360ª funciona de la misma forma que un sensor laser convencional, como hemos explicado anteriormente, pero con la diferencia que tiene un cabezal giratorio. En este proyecto se ha utilizado un modelo preconfigurado de Gazebo de un sensor laser de distancia de la marca comercial Hokuyo, este se muestra en la *Imagen 10*.

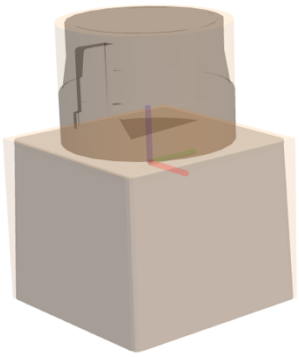


Imagen 10 Diseño del sensor

Este modelo no es igual al que necesitamos ya que no tiene un cabezal móvil y también se tiene que modificar el láser ya que en el modelo incluido en Gazebo, se incluyen un conjunto de láser que forman un plano. En nuestro proyecto necesitamos un único laser con un cabezal giratorio e incluirlo en el modelo de la *Imagen 10*.

Gazebo utiliza SDF para generar un modelo, Para el cambio se necesita modificar el sensor para que se componga de un único haz de luz y cumpla las especificaciones que se desea simular, en este caso serían: distancia mínima de 0.3 metros, y distancia máxima de 40 metros con una resolución de 0.1 metros. También se necesita añadir una unión giratoria entre la base del sensor y la parte superior donde se aloja el sensor laser, la velocidad de giro se puede configurar para adaptar la velocidad a la realizada por un sensor real que en el caso de este proyecto se ha configurado a una velocidad de 150 rpm. Una vez el modelo está modificado se ha incluido en la parte superior del cuadricóptero como podemos ver en la *Imagen 11*.

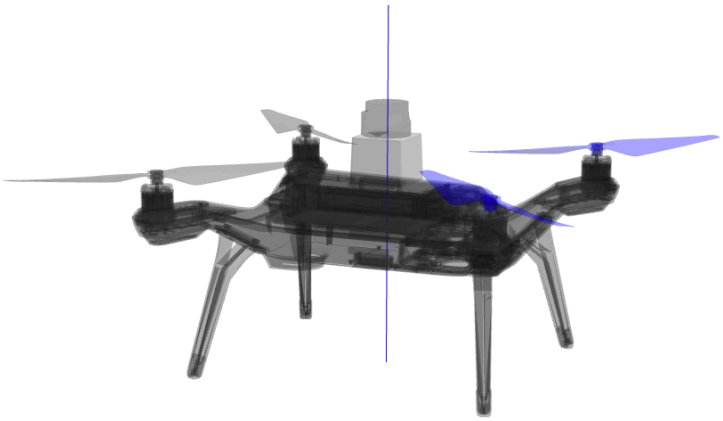


Imagen 11 Diseño completo

También ha sido necesario crear un nuevo sistema de control del sensor, ya que tenemos que tener en cuenta la posición de la parte superior donde se encuentra el sensor laser. Para ello debemos obtener la información de la posición del sensor expresada en cuaternios, que son una extensión de números reales que nos permiten representar de un cuerpo sólido tanto la orientación como la posición del mismo, Esta representación se basa en un punto, que normalmente será la base del robot. En nuestro caso tomamos la referencia del sensor respecto a la base del mismo. A partir del cuaternio se puede obtener los ángulos de Tait-Bryan, que se compone de tres ángulos que definen la rotación de un triángulo respecto a un punto de referencia. En la *Imagen 12*, podemos ver los ángulos que componen Tait-Bryan, de los cuales se necesitan el ángulo Yaw para poder conocer cuál es la dirección donde se encuentra el objeto detectado.

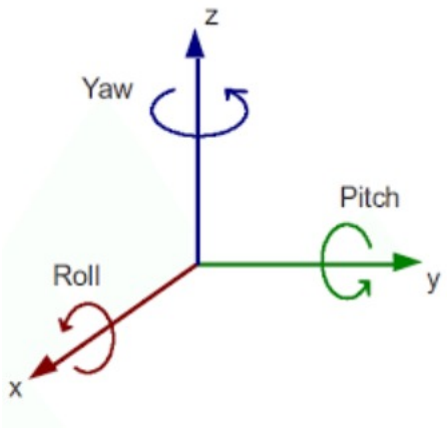


Imagen 12 Ángulos de aeronave

Los cuaternios se representan de la siguiente forma: $q = [q_0, q_1, q_2, q_3]$ Para poder obtener los ángulos de Tait-Bryan, se utiliza la siguiente ecuación:

$$\begin{bmatrix} Roll \\ Pich \\ Yaw \end{bmatrix} = \begin{bmatrix} arctan\dfrac{2(q_0q_1 + q_2q_3)}{1 - 2(q_1^2 + q_2^2)} \\ arcsin\big(2(q_0q_2 - q_1q_3)\big) \\ arctan\dfrac{2(q_0q_3 + q_1q_2)}{1 - 2(q_2^2 + q_3^2)} \end{bmatrix}$$

El ángulo obtenido a través de esta ecuación se mide en radianes, Para tratar de forma más sencilla la información obtenida del sensor y para utilizar los mensajes preconfigurados de Mavlink se necesita utilizar números enteros, con radianes el error producido por usar números enteros es demasiado grande por lo que se utilizarán grados donde el error al utilizar grados es más pequeño.

Después de obtener el ángulo y la distancia a la que se encuentra el obstáculo, se encapsula en el mensaje predefinido por Mavlink y se envía la información al PX4. Después de analizar la información recibida en PX4 se pudo comprobar que la frecuencia a la que trabaja Mavlink no es suficiente como para enviar la información del sensor de distancia correctamente.

Se pudo comprobar que modificando la velocidad de la parte superior del sensor se conseguía una frecuencia de unos ochenta grados en cada lectura. Para solucionar este problema fue necesario sacrificar precisión en el sensor, pero el lector sigue siendo igual de seguro enviando información importante, para ello utilizamos el siguiente sistema:

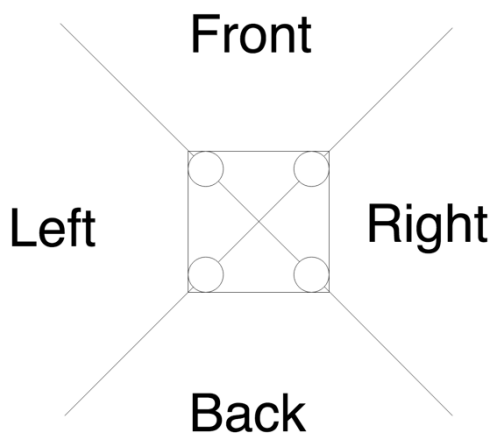


Imagen 13 Esquema división de medidas

Como podemos ver la *Imagen 13*, cada noventa grados se envía información del sensor, en este caso se envía la distancia mínima obtenida en el intervalo correspondiente. Como hemos explicado anteriormente, la información se envía con una frecuencia de ochenta grados, por lo que cada nueve envíos, el sensor repetirá información, lo cual no es un problema. Con este sistema no se tendrá una precisión tan grande como cuando si se enviarán todos los ángulos con su información, pero se sigue pudiendo detectar de una forma bastante efectiva si existe peligro en la dirección que lleva el vehículo.

3.4. Evasión de obstáculos

Como se ha explicado anteriormente, PX4 es un sistema basado en un sistema operativo de tiempo real que funciona a través de tareas, y que existen tareas secundarias que se realizan en segundo plano sin interferir en el correcto funcionamiento del controlador de vuelo. El primer paso que se debe realizar es crear una nueva tarea secundaria, en un principio la tarea se creará con los valores por defecto del sistema: el tipo de planificador, la prioridad y el tamaño de la pila.

Después de crear la tarea, ya se puede empezar a programar el sistema de evasión; empezaremos por suscribirnos a los mensajes uORB necesarios, en este caso se empezará por el sensor de distancia, la posición local del vehículo donde se encuentran datos de velocidad del vehículo para calcular la distancia de seguridad y el modo en el que se encuentra el vehículo, ya que este módulo solo debe funcionar en el caso de que el dron esté en una misión.

En el mensaje de uORB la velocidad viene como un vector de tres posiciones donde se indica la velocidad en cada uno de los sentidos. A continuación, en la *Imagen 14*, podemos ver el desglose de las velocidades.

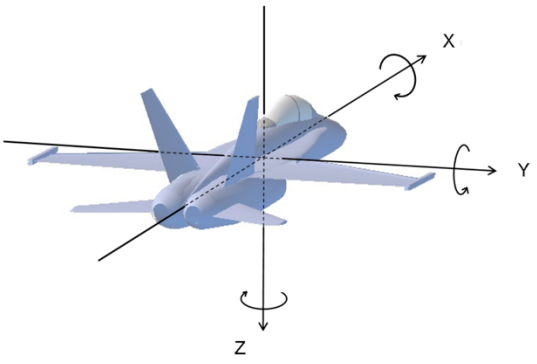


Imagen 14 Ángulos de aeronave en cartesianas

La velocidad de descenso no es relevante en este proyecto, ya que el objeto lo detectamos en el mismo donde se encuentra el dron, es decir, en el plano X e Y. Únicamente necesitamos calcular la velocidad de desplazamiento en ese plano y como está dividida en dos vectores, se necesita calcular el módulo de ambos. Para este cálculo se utilizará el teorema de Pitágoras que dice que la hipotenusa al cuadrado es igual a la suma de los catetos al cuadrado, transformado a formula sería: $h^2 = a^2 + b^2$ donde h sería la longitud de la hipotenusa del triángulo y a y b la longitud de los catetos del triángulo. En el caso de las velocidades, a y b serían la velocidad de X y la velocidad de Y, y h será el módulo de ambas y por tanto la velocidad real del dron.

A partir de la velocidad del dron, se calcula la distancia de seguridad, esta distancia de seguridad será la utilizada por el sistema de evasión para empezar las maniobras o cambiar entre las distintas fases del sistema que se explicarán a continuación.

Para el cálculo de la distancia de seguridad se han realizado un conjunto de pruebas simuladas para obtener datos sobre la capacidad de reacción del dron a distintas velocidades, el dron simulado para este conjunto de pruebas es el mismo vehículo que se utilizará posteriormente en las pruebas finales del proyecto. Esta distancia de seguridad está basada en los datos obtenidos en la simulación, en el caso de utilizar este sistema en vuelos reales o en otros simuladores, la distancia de seguridad debe ser adaptada al entorno.

Los datos obtenidos en la siguientes gráficas se obtienen a partir de una herramienta que proporciona DroneCode para el análisis de un vuelo, [15] tanto simulado como real, la velocidad está descompuesta en las distintas coordenadas, en este caso la altura es la única unidad que no se modificará por lo que se ha decidido ocultarla, las velocidades utilizadas para la obtención de las gráficas son: 5 m/s, 7 m/s y 8.1 m/s. La velocidad máxima es de 8.1 m/s por lo que, si el obtenemos el resultado de la velocidad máxima, para el resto de las velocidades debería ser menor, para ello se han añadido las velocidades de 5 y 7 m/s.

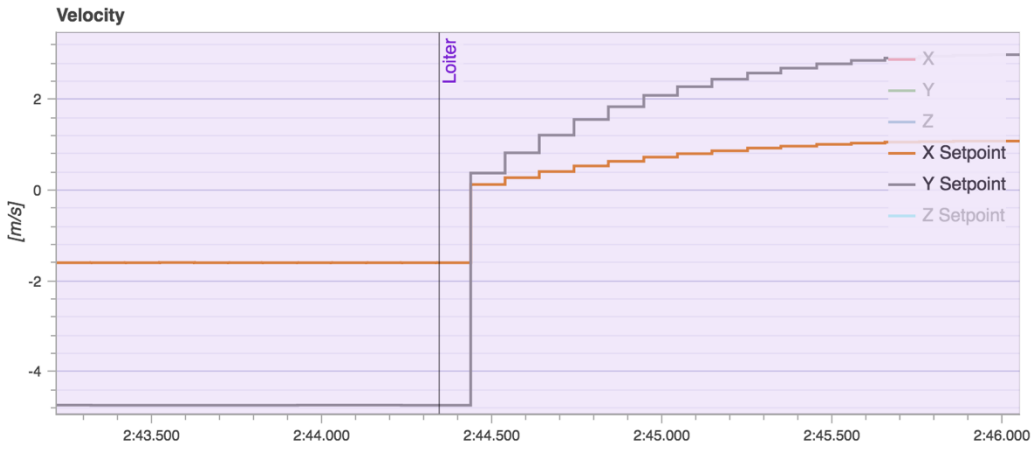


Imagen 15 Velocidad 5 m/s

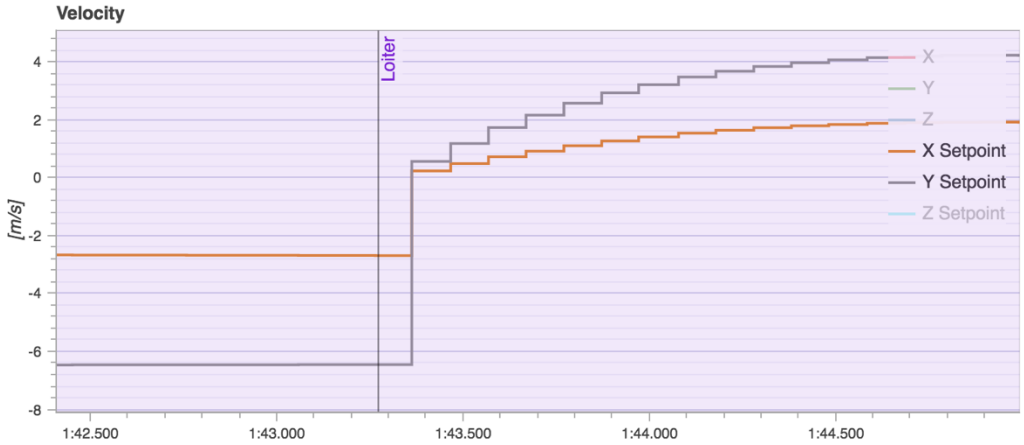


Imagen 16 Velocidad 7 m/s

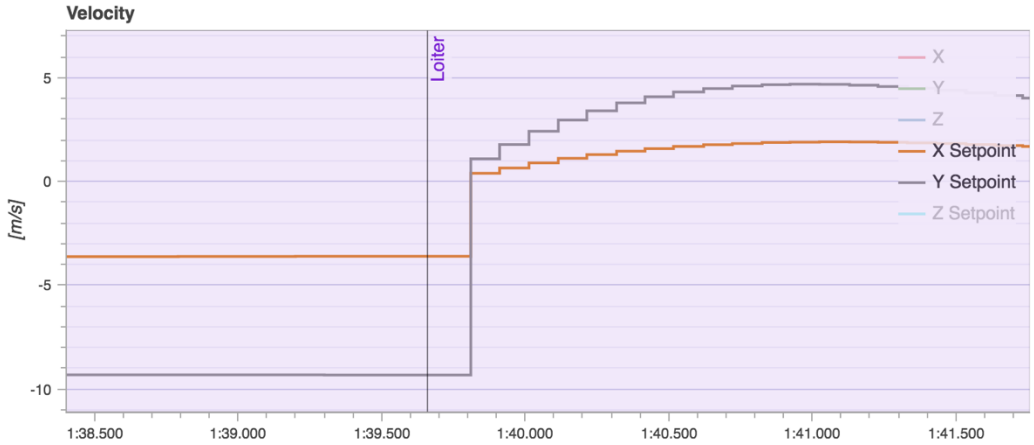


Imagen 17 Velocidad 8.1 m/s

Como podemos ver en las gráficas anteriores la capacidad de reducir la velocidad a 0 m/s es muy alta y no tarda más de 200 ms en el caso de velocidad más alta *Imagen 17* se ha decidido añadir un segundo como sistema de seguridad.

A parte de la capacidad de frenado se deberá añadir una distancia de seguridad que complementa la velocidad de reacción del dron, para ello también se han realizado diferentes pruebas y en ningún caso se ha obtenido una capacidad de reacción mayor a un segundo, por lo que la distancia de seguridad será la siguiente:

$$DS = 2 * \sqrt{v_x^2 + v_y^2}$$

Para hacer un sistema de detección funcional, aunque no es el objetivo principal de este proyecto, se hará un sistema de detección que consta de tres fases principales: No peligro, peligro y evasión. A continuación, se describirá cada una de las fases.

3.4.1. No Peligro

En esta fase únicamente se comprobará la distancia del sensor y se comparará con la distancia de seguridad. Si es menor la distancia del sensor, entonces el sistema entrará en la fase de peligro.

3.4.2. Peligro

Esta fase está creada para evitar falsas lecturas ya que la probabilidad de dos falsas lecturas es muy baja. Lo que, sí es probable y común, es que la distancia leída por el sensor sea la distancia del dron al suelo y por lo tanto un error.

Para evitar las lecturas del suelo se ha ideado un sistema que consiste en calcular la altura a la que se encuentra el objeto detectado, Si la altura calculada, es la misma, con un margen de error, a la altura real del dron, significa que lo que se está leyendo es el suelo y no un obstáculo, En las imágenes *Imagen 18 y Imagen 19,* podemos ver un ejemplo del sistema, en la *Imagen 18* se puede ver una lectura correcta de un objeto y en la *Imagen 19* se puede ver una lectura falsa del suelo.

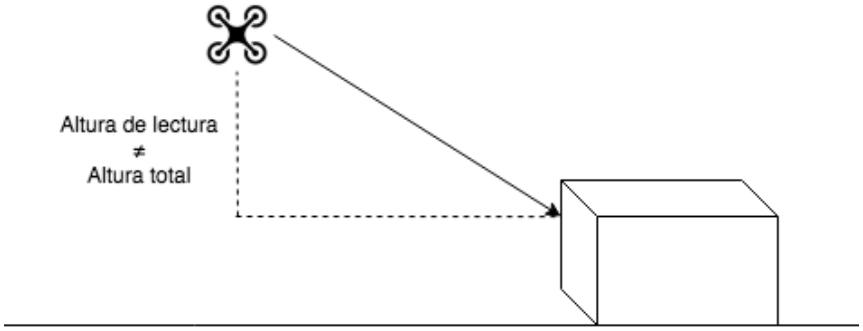


Imagen 18 Esquema detección obstáculo

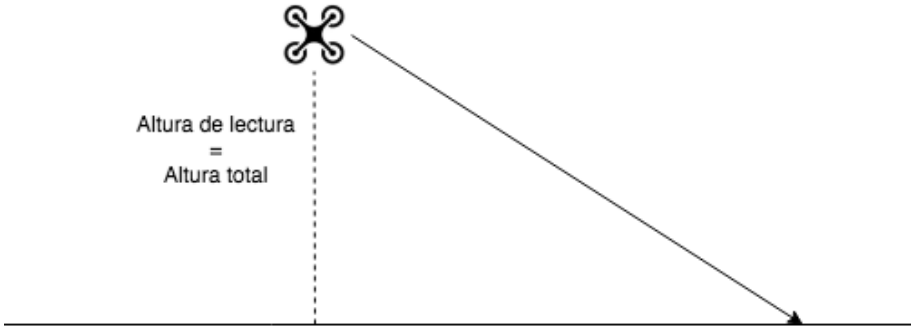


Imagen 19 Esquema lectura del suelo

Para calcular la altura se utilizará geometría básica y con la siguiente fórmula se podrá calcular la altura desde el dron al punto de lectura: $h = \cos\alpha * d$, donde h es la altura que deseamos calcular, y α es el ángulo de inclinación del dron, En la *Imagen 14*, se puede ver el ángulo que se utilizará para la inclinación delantera del dron, el ángulo Pitch, Por último, d es la distancia obtenida por el sensor de distancia. En la *Imagen 20*, se muestra un esquema visual del cálculo.

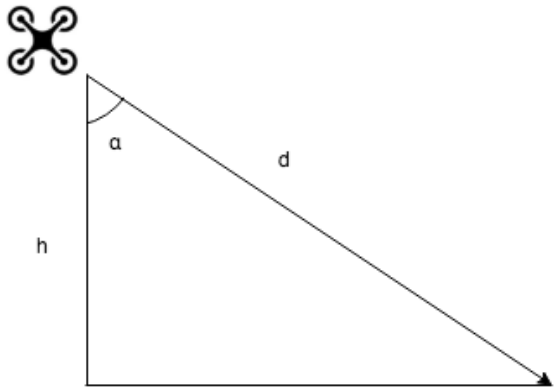


Imagen 20 Esquema cálculo de altura

Después de evitar las lecturas del suelo, se procederá a volver a comprobar la distancia del sensor con la distancia de seguridad, En el caso de que la distancia del sensor sea menor, el sistema pasará a la fase de evasión. En caso contrario de que la distancia del sensor sea mayor que la distancia de seguridad indicaría que no es necesario una evasión y después de comprobar tres veces este dato, el sistema volvería a la fase de no peligro.

3.4.3. Evasión

Para evitar los posibles obstáculos se utilizará el comando de pausa implementado en el sistema, Si a este comando se le añade unas coordenadas y una altura, el dron irá hasta dichas coordenadas y altura, y en ese momento se detendrá. La altura, como se ha explicado anteriormente, será la misma a la que se encuentre el dron ya que el sensor solo puede detectar objetos en su misma altura, Se deberán añadir al comando dos nuevas coordenadas perpendiculares al próximo objetivo de la misión. A continuación, se explicará el cálculo de estas coordenadas. Una vez el dron llega al objetivo para la evasión, su velocidad se detendrá y será muy cercana a 0, en ese momento, continuaremos la misión, y el sistema volverá a la fase de no peligro.

Para calcular las nuevas coordenadas se utiliza nuevamente cálculos geométricos, en este caso vectoriales, siendo el primer cálculo por realizar para obtener el vector que une el dron con el objetivo actual de la misión. En la *Imagen 21*, podemos ver los dos puntos que se usarán en el cálculo, la posición del dron, $P_o = (x_o, y_o)$ y el punto de destino de la misión, $P_f = (x_f, y_f)$, estos puntos se usarán para obtener el vector que indica la dirección de movimiento del dron. El cálculo necesario es: $v = (x_f - x_o, y_f - y_o)$,

Como se necesita un punto perpendicular al vector que se ha hallado anteriormente, también se necesita un vector normal, un vector es normal a otro vector si su producto escalar es igual a cero. Invirtiendo el orden de los componentes del vector y cambiando uno de signo, se obtiene siempre el vector normal, a continuación, demostraremos esta operación.

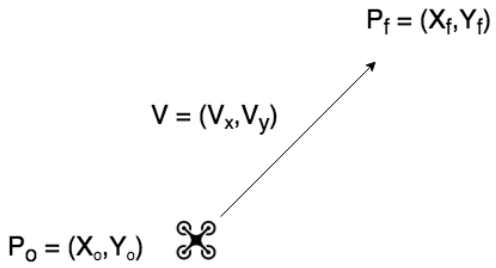


Imagen 21 Vector entre dos puntos

El producto escalar de dos vectores: $v = (x_v, y_v)$ y $n = (x_n, y_n)$ es: $p = (x_v x_n + y_v y_n)$.

Si nombramos el vector que une el punto de destino de la misión y la posición del dron, $v = (a, b)$, el vector normal será $n = (b, -a)$ o $n = (-b, a)$, Dependiendo de la posición del signo, el sentido del vector será uno u otro. El vector normal podemos demostrarlo fácilmente si hacemos el producto escalar: $0 = \big(ab + a(-b)\big)$. En la *Imagen 22*, podemos ver un diagrama del cálculo.

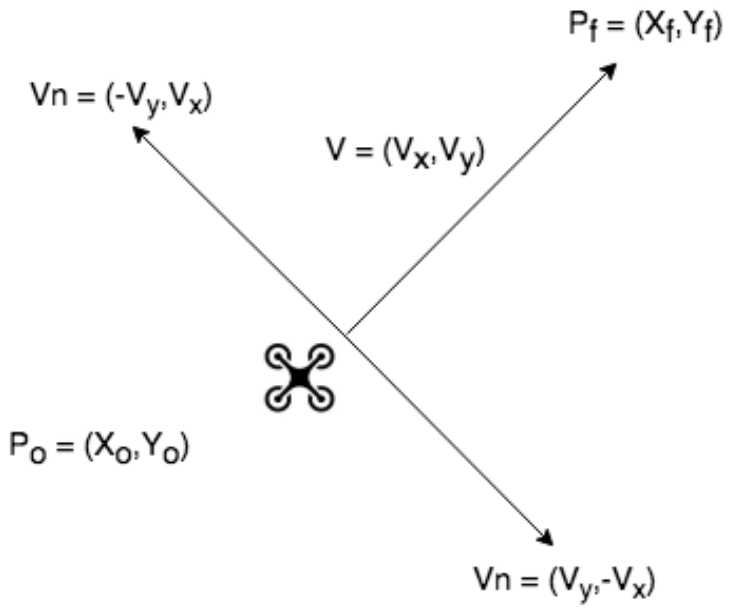


Imagen 22 Vector normal a otro vector

Para calcular el punto perpendicular utilizamos el proceso inverso al anterior, partiendo del vector normal y de la posición de origen se puede obtener el nuevo punto, pero primero se necesita escalar el vector normal a la distancia deseada entre ambos puntos. Una vez, el vector normal esta escalado, usaremos el procedimiento de obtener un vector a partir de dos puntos. $v_e = (x-x_o, y-y_o) \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} x - x_o \\ y - y_o \end{bmatrix}$, si despejamos las variables que conocemos, obtendremos la siguiente formula $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} V_x + x_o \\ V_y + y_o \end{bmatrix}$, con esta fórmula se puede obtener las coordenadas que se necesitan para esquivar el objeto. A continuación, se mostrará una ejecución del sistema.

Para decidir si se debe esquivar a la izquierda o la derecha, el dron antes de iniciar la maniobra de evasión, analizará los sectores izquierdo y derecho, como podemos ver en la imagen A, si no hay ningún obstáculo a la izquierda, el dron esquivará a la izquierda, si hay algún obstáculo el dron mirará posibles peligros a la derecha, al igual que en la izquierda, si hay no hay peligro esquivará a la derecha, pero si hay peligro esquivará hacia la parte trasera del dron, y se repetiría el proceso. Si por algún motivo el dron corre peligro en todas direcciones, automáticamente aterrizará.

4. PRUEBAS

En este apartado se abordarán las pruebas realizadas después de realizar la implementación total del sistema, para analizar las pruebas se va a utilizar una herramienta proporcionada por PX4: *https://logs.px4.io*. Esta herramienta permite leer unos ficheros log generados por QGroundControl que contienen mucha información del dron durante la misión que ha realizado.

Para la realización de las pruebas se ha decidido realizar una misión sin obstáculo y posteriormente la misma misión con obstáculos, poniendo al dron en situaciones varias situaciones donde tenga que esquivar uno o varios objetos, que en este caso serán casas como se verá a continuación.

Para empezar, se va a realizar una misión sencilla de un único punto el dron deberá despegar e ir hacía el punto 1 sin detenerse y esperar allí nuevas ordenes. En la *Imagen 23* podemos ver la misión que debe realizar.

Imagen 23 Misión utilizada en pruebas

En esta misión incluiremos una casa en el camino que debe realizar el dron como podemos ver a continuación en la *Imagen 24*. La imagen representa el momento exacto donde el dron detecta la casa como un obstáculo e inicia la evasión.

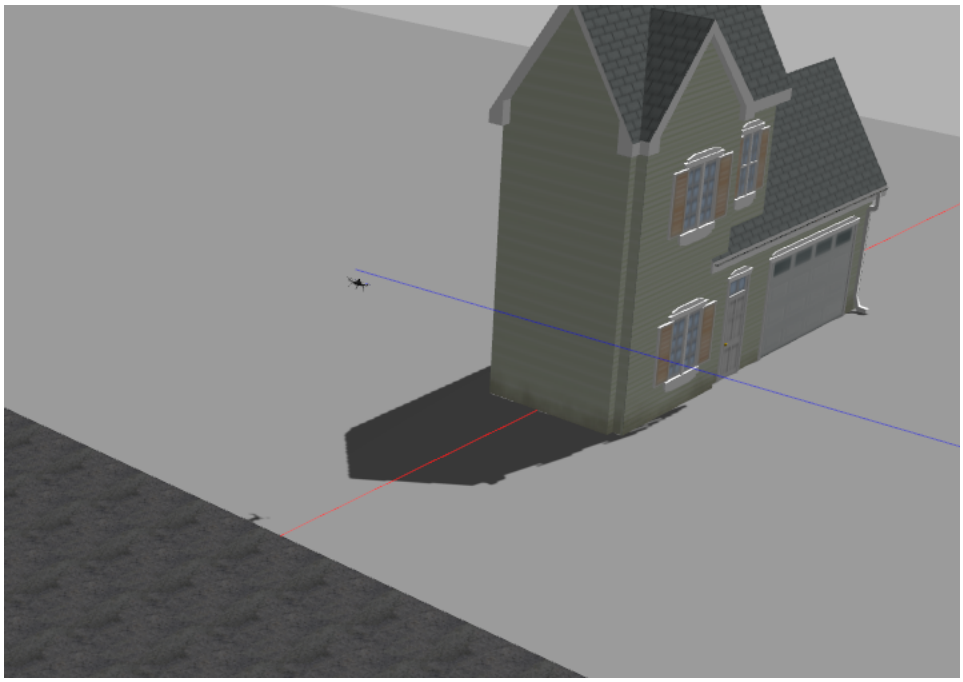


Imagen 24 Dron detectando obstáculo

A continuación, se va a realizar una comparación de ambas misiones, primero sin obstáculo y después con el obstáculo y la maniobra de evasión. Primero vamos a analizar una comparativa de la respuesta obtenida en QGroundControl en ambas misiones.

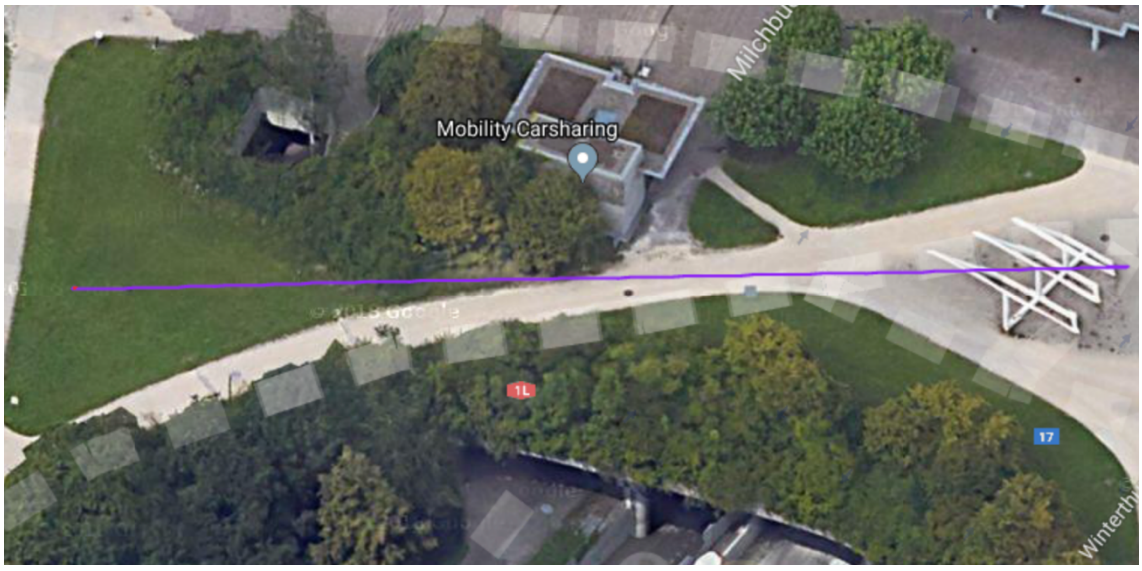


Imagen 25 Resultado de la prueba misión sin obstáculo

Imagen 26 Resultado de la prueba con obstáculo

Como podemos ver en la *Imagen 25*, podemos ver como la misión se realiza sin ningún giro, el dron va hacia el punto sin desviarse en el camino. En la *Imagen 26*, se puede ver como el dron realiza un giro de 90º para evitar el obstáculo y posteriormente se dirige de hacía el objetivo.

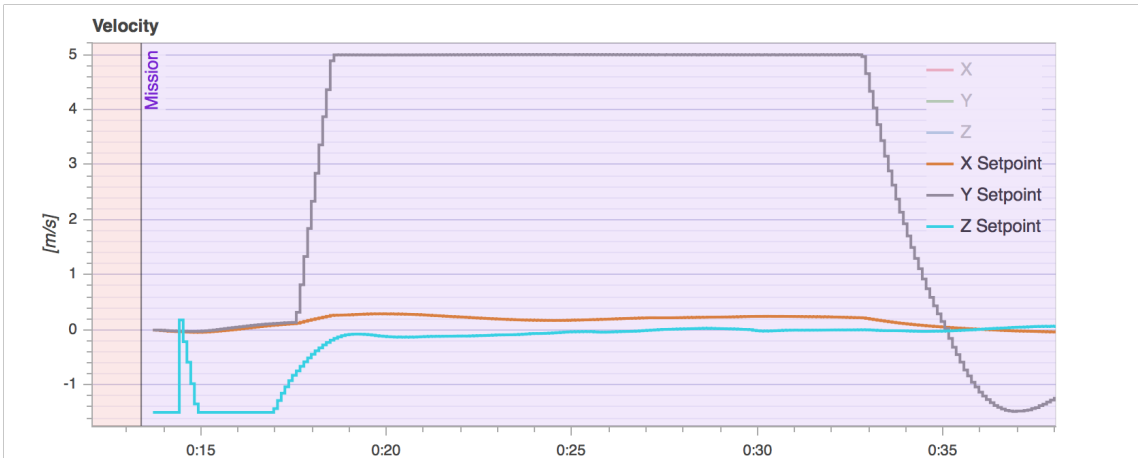


Imagen 27 Velocidad sin obstáculo

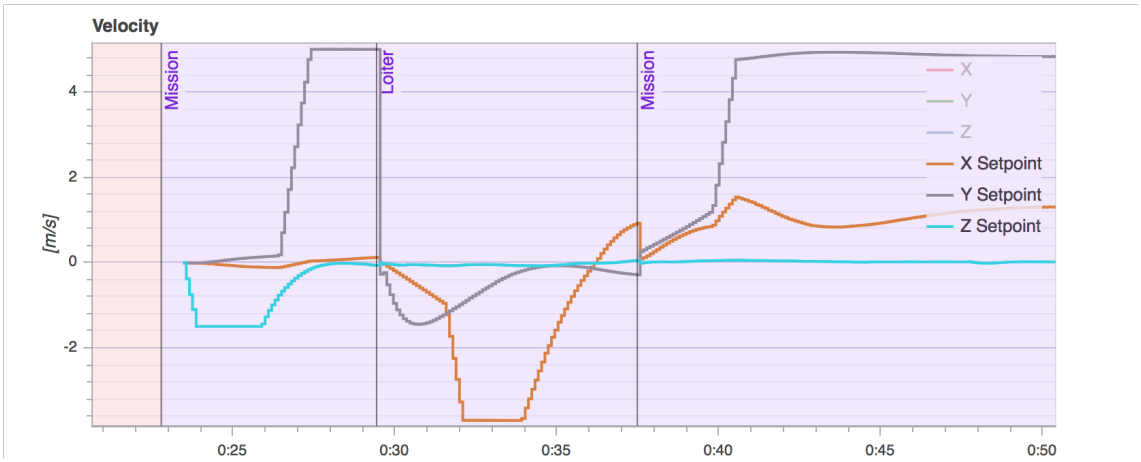


Imagen 28 Velocidad con obstáculo

En las *Imagen 27 y Imagen 28* anteriores podemos ver como en el caso donde no hay obstáculo las velocidades son constantes desde que empieza la misión y el dron despega hasta que termina la misión, en esta gráfica la velocidad esta descompuesta en coordenadas cartesianas, si nos fijamos en la función z vemos que solo se modifica en el inicio de la misión para ascender el dron y el resto de misión se mantiene constante, después la función x e y, dependen de la dirección de la misión, en este caso ha dado la casualidad que la misión está únicamente en la dirección de y, como se puede apreciar en la *Imagen 27*. En la *Imagen 28* podemos ver que cuando detecta el objeto todas las velocidades bajan a 0 o incluso son negativas para contrarrestar la frenada, posteriormente la descomposición de la velocidad en X e Y, cambia como podemos ver en la *Imagen 26*. En la siguiente gráfica se puede ver el momento en el que el sensor de distancia detecta un objeto.

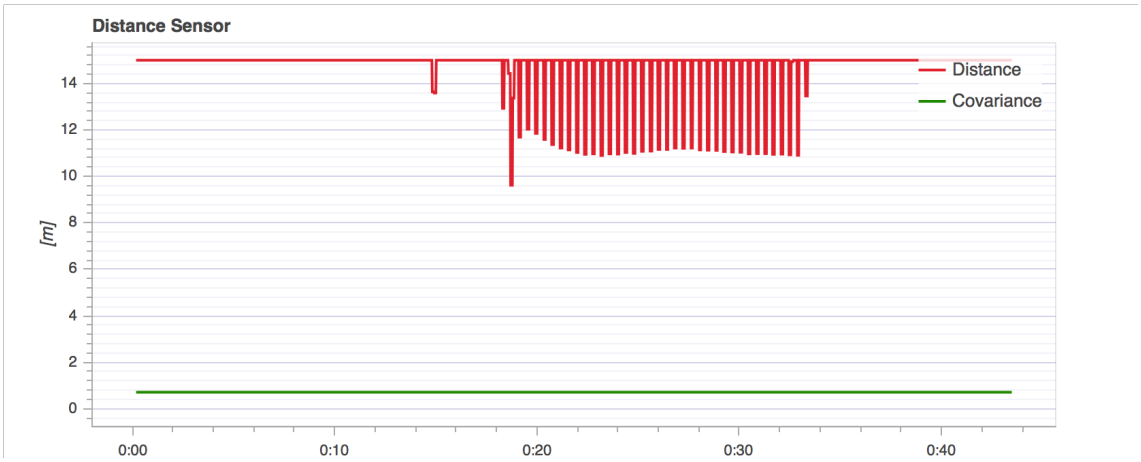


Imagen 29 Sensor de distancia sin obstáculo

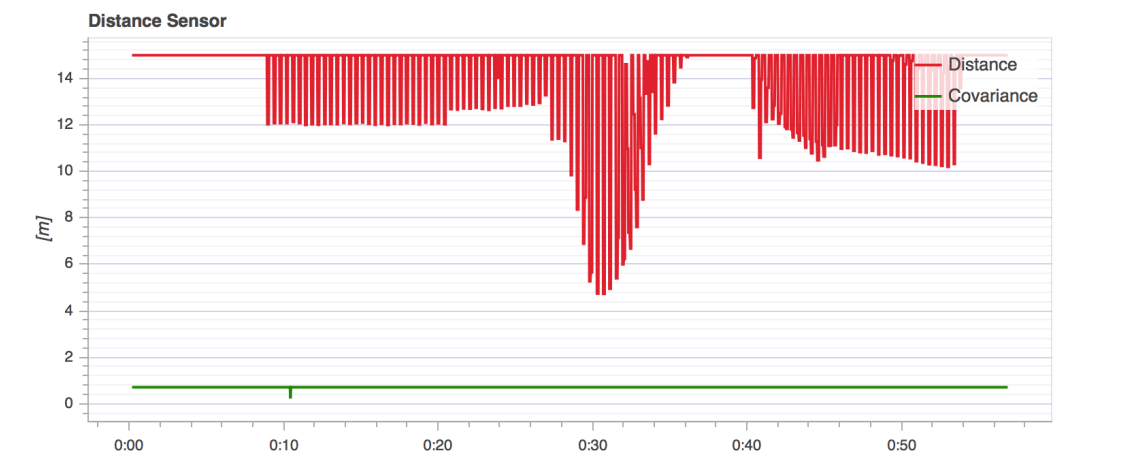


Imagen 30 Sensor de distancia con obstáculo

En las *Imagen 29 y Imagen 30* se puede ver que el sensor de distancia detecta el objeto en el segundo 30 aproximadamente, en este momento es cuando el dron frena y ejecuta la maniobra de evasión, en la *Imagen 29* que sería la que no hay obstáculo hay varias lecturas, estas lecturas son producidas por ruido en el sensor o por la detección del suelo como un obstáculo, por este motivo se tubo que tener en cuenta este problema de la detección del suelo a la hora de ejecutar la maniobra de evasión ya que como se ve claramente en el ejemplo, la distancia de seguridad sería de 10 metros y hay una lectura de más de 10 metros lo cuál habría ejecutado el sistema de evasión.

En conjunto de la *Imagen 28 y Imagen 30*, podemos observar que la maniobra de evasión empieza cuando el dron detecta una distancia menor de 10 m, no es exactamente en ese momento por problemas de latencia, aunque se acerca mucho a este valor, que sería la distancia de seguridad calculada en el apartado *3.4. Evasión de obstáculos.*

A continuación, vamos a ver como el dron reacciona a la evasión primero la inclinación que utiliza para frenar, en este caso el ángulo que mide este parámetro es el pich y posteriormente el giro que realiza de forma horizontal de 90º a través del ángulo llamado yaw.

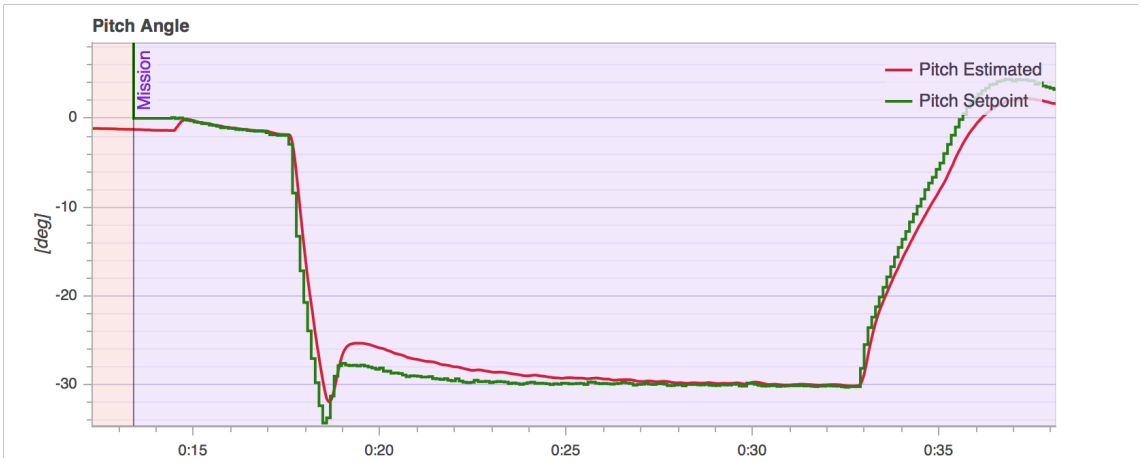


Imagen 31 Pich sin obstáculo

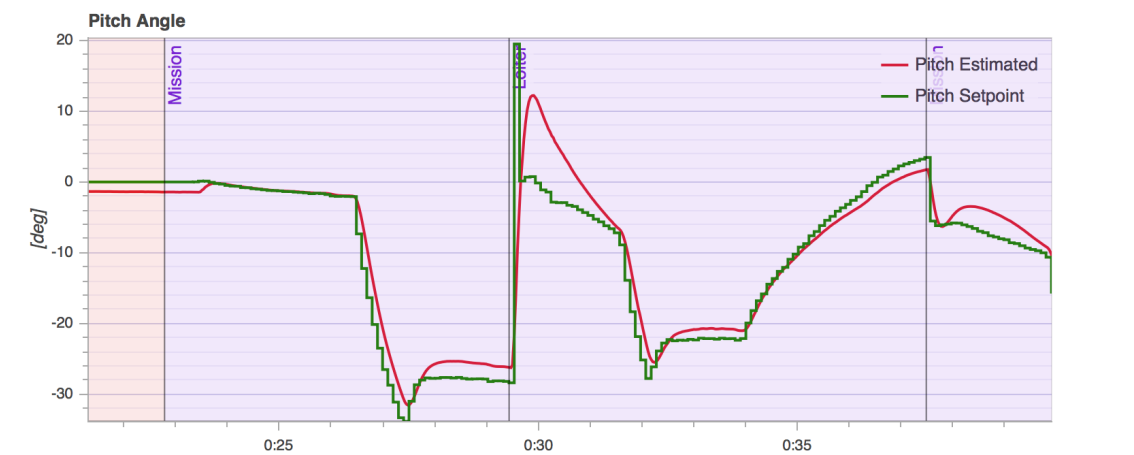


Imagen 32 Pich con obstáculo

En las *Imagen 31 y Imagen 32* se puede ver que cuando empieza la misión en ambas gráficas se puede ver que hay unos segundos donde no se mueve el pich, esto es porque el dron está despegando. Posteriormente en la *Imagen 32* se inclina hacía adelante para moverse y cuando empieza la maniobra de evasión el pich aumenta a positivo para frenar el dron y se estabiliza en 0, después aumenta de nuevo para evitar el objeto, una vez esquivado el objeto, vuelve a bajar a 0 para girar al objetivo de la misión y se vuelve a inclinar hacia delante para avanzar y terminar la misión. Mientras que en la *Imagen 31*, el pich no se modifica hasta que éste termina la misión

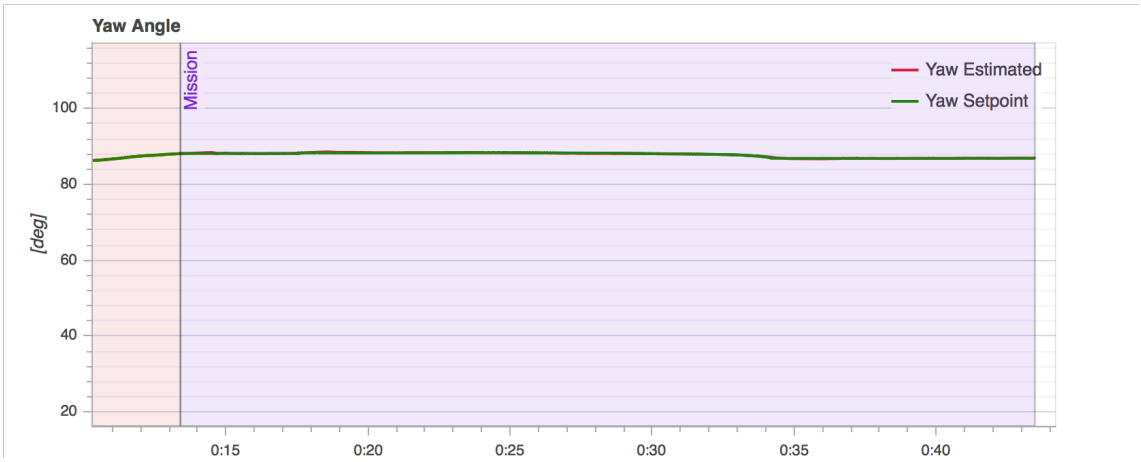


Imagen 33 Yaw sin obstáculo

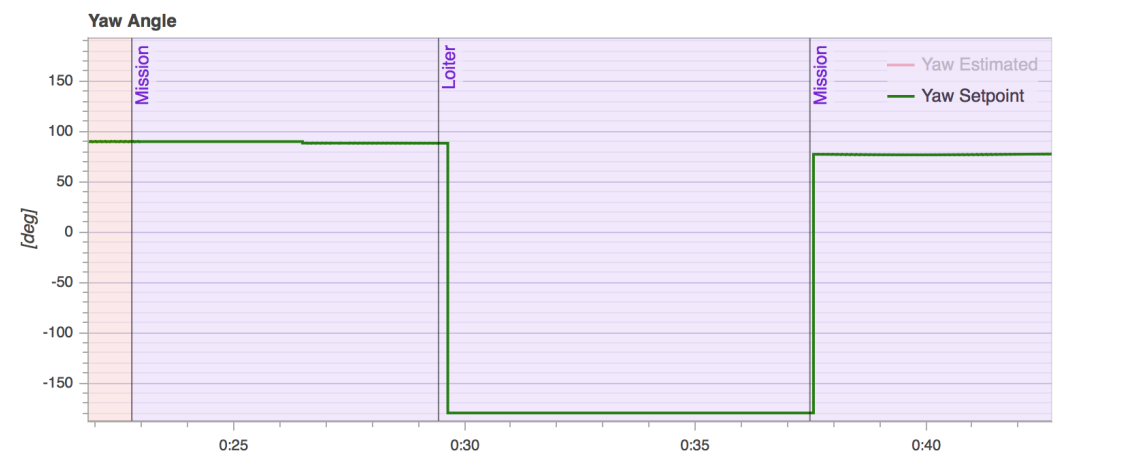


Imagen 34 Yaw con obstáculo

En la *Imagen 34* anterior se puede ver como el dron realiza 90º para girar de forma horizontal una vez empieza la maniobra de evasión, en un principio el dron lleva una dirección de 90º, para esquivar debería aumentar 90º que son 180º, aunque esto se indica en la gráfica como -180º. En la *Imagen 33* sin obstáculo el yaw se mantiene estable durante toda la misión.

5. MARCO LEGAL

En este apartado se explicarán todas las cuestiones referentes a la legislación que afecta al proyecto, Este proyecto tiene dos partes legales diferenciadas, la primera hace referencia a las licencias de las herramientas utilizadas, y la segunda parte hace referencia al apartado legal dentro de España para el vuelo de vehículos aéreos autónomos.

Todas las herramientas utilizadas en este proyecto son herramientas de código abierto, a continuación, se irá analizando una por una el tipo de licencia de las herramientas. Que una herramienta sea de código abierto significa que se podrá utilizar de forma gratuita, también se ofrece el código fuente para realizar modificaciones, pero la mayoría de las herramientas constan de una licencia para administrar apropiaciones indebidas o la mala utilización de la herramienta.

5.1. PX4

La licencia que incorpora PX4 es “3-Clause BSD License”, esta licencia se encarga de asegurarse que la redistribución del código y del binario están sujetos al copyright del producto, también se encarga de asegurar el nombre de la herramienta, de los contribuidores y del propietario del copyright para promociones sin permiso.

5.2. Mavlink

Mavlink contiene la licencia “Lesser General Public License version 3” que se encarga de proteger el sistema con los derechos de autor originales. Esta licencia tiene la particularidad de que puede ser usada como librería dentro de un programa y mantener la licencia junto a la nueva licencia del programa que la utiliza.

5.3. GAZEBO

Gazebo incorpora una licencia de Apache, concretamente la versión 2.0, la cual conserva los derechos de autor y el descargo de responsabilidad, aunque esta licencia no requiere de la redistribución del código fuente después de una modificación del mismo.

5.4. NuttX

NuttX incorpora una licencia de BSD (Berkeley Software Distribution), que es una licencia de código libre muy permisiva, estando cerca del dominio público. La licencia BSD permite el uso del código fuente en software no libre, aunque en este caso al estar incluido dentro de un software libre, esta característica no aplica. [16]

5.5. Ubuntu

Ubuntu trabaja mediante su propia licencia de código abierto que permite su uso para disfrute personal o para organizaciones y la libre modificación del código, aunque no se pueda distribuir dichas modificaciones, ya que, para poder ser distribuidas, estas modificaciones deben ser aprobadas por Canonical. En el proyecto actual, se utilizará la versión original de Ubuntu bajo la versión 14.04.

5.6. Ley española para drones

La ley de española para vuelos de vehículos no tripulados está en continuos cambios ya que es un sector que avanza a gran velocidad. Actualmente el vuelo de vehículos aéreos no tripulados tiene que ser siempre bajo el mando de un piloto, lo cual supone un gran inconveniente para el sistema desarrollado en el proyecto ya que actualmente no tendría ningún uso real por problema legales. Si hay un piloto detrás del dron, en zonas despobladas el campo de visión del dron es de 500 metros, una superficie óptima, siempre que el vuelo se realice a más de 8 kilómetros de un aeropuerto o zona de aviación, con una altura máxima de 120 metros. En zonas urbanas, la ley es más estricta y permite vuelos con una visualización de 100 metros, a más de 50 metros de edificios y personas. Si el vuelo genera un posible riesgo para personas o bienes, entonces se deberá pedir permiso a la Agencia Estatal de Seguridad Aérea.

6. ENTORNO SOCIOECONÓMICO

Este apartado está dedicado a explicar toda la información relacionada con el apartado económico, se mostrará el presupuesto inicial y posteriormente se mostrará el coste real del proyecto. Debido a las complicaciones surgidas durante el proyecto, se podrá ver si el cálculo inicial ha sido correcto.

6.1. Desarrollo del proyecto

El proyecto tiene una carga de trabajo de 12 créditos ECTS, según la Universidad Carlos III de Madrid, un crédito ECTS equivale a 25 horas de trabajo, por lo que el proyecto debe tener una duración aproximada de 300 horas de trabajo. Esta cifra ha sido superada como podemos ver a continuación:

APARTADO	TAREA	INICIO	FIN	HORAS
ESTADO DEL ARTE	Estudio de simuladores	06/11/2017	27/11/2017	30
	Estudio de Mavlink	27/11/2017	04/12/2017	8
	Estudio de NuttX	04/12/2017	08/12/2017	6
	Estudio del sensor	08/12/2017	15/12/2017	8
	Estudio de PX4	15/12/2017	24/01/2018	60
INSTALACIÓN	Instalación del sistema DroneCode	15/12/2017	24/01/2018	20
DESARROLLO	Diseño del sensor	24/01/2018	12/02/2018	8
	Programación del controlador del sensor	12/02/2018	22/02/2018	15

	Programación tarea en PX4	22/02/2018	02/03/2018	5
	Diseño sistema de evasión	02/03/2018	12/03/2018	10
	Programación del sistema de evasión	12/03/2018	28/05/2018	80
PRUEBAS	Comprobar funcionamiento del sensor	28/05/2018	14/06/2018	10
	Comprobar funcionamiento de la tarea en PX4	14/06/2018	10/07/2018	13
	Comprobar funcionamiento del sistema de evasión	10/07/2018	08/08/2018	11
DOCUMENTACIÓN	Documentación	22/06/2018	29/08/2018	130
REVISIÓN	Revisión	03/09/2018	21/09/2018	20
	Horas totales			434

Tabla 30 Planificación del trabajo

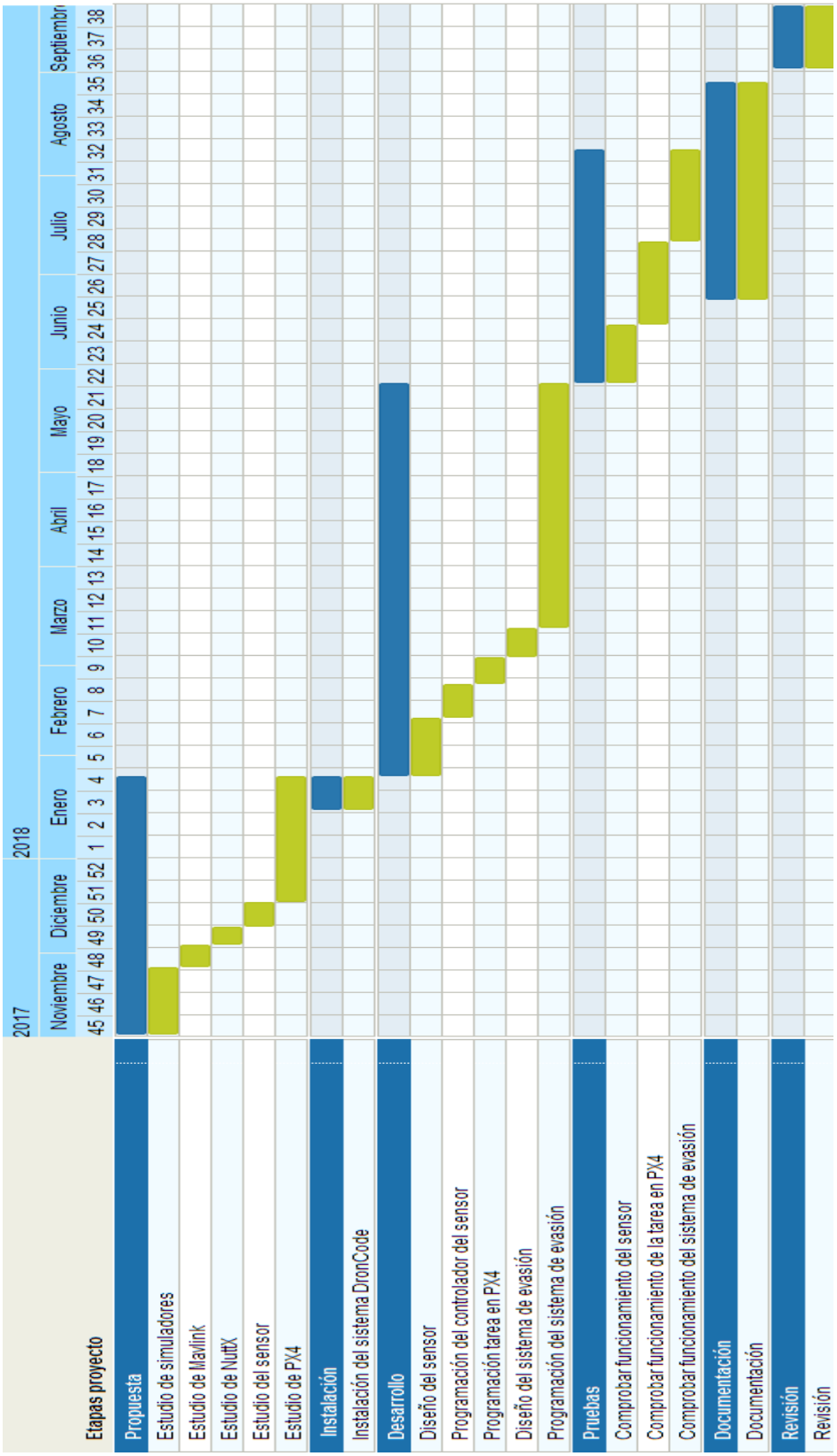


Imagen 35 Diagrama de GANT

6.2. Presupuesto

En este apartado se analizará el presupuesto para la realización del proyecto. Se empezará por el coste material del proyecto, se continuará con el coste personal del proyecto para terminar con el coste total del proyecto.

6.2.1. Coste material

El coste material de este proyecto se ha basado en la utilización de un ordenador, en este caso un macBook pro 2017 con un precio de 1350 €, la duración del proyecto ha sido aproximadamente de unos nueve meses, y el periodo de depreciación del ordenador está entorno a 48 meses, esto genera un coste de 254€.

6.2.2. Coste personal

El coste de personal se basará en dos personas, el jefe de proyecto, Jesús García Herrero y el analista y programador, Enrique Martínez Martínez. A continuación, se mostrará en la *Tabla 31* con las horas dedicadas por cada miembro del proyecto:

ACTIVIDAD	**HORAS**	
	Analista y programador	Jefe de proyecto
ESTADO DEL ARTE	112	10
INSTALACIÓN	20	0
DESARROLLO	118	0
PRUEBAS	34	0
DOCUMENTACIÓN	130	0
REVISIÓN	20	40
TOTAL	434	50

Tabla 31 Horas detalladas

El precio por hora se ha calculado en base a el salario mínimo y máximo de los datos oficiales de la Seguridad Social en la categoría de Ingeniero Técnico y Licenciado [17]. En la siguiente *Tabla 32* se mostrarán todos los datos del coste de personal.

	ANALISTA PROGRAMADOR	JEFE DE PROYECTO
HORAS	434	50
COSTE POR HORA	19,00 €	30,00 €
COSTE TOTAL POR CATEGORÍA	8.246,00 €	1.500,00 €
COSTE TOTAL	9.746,00 €	

Tabla 32 Coste de horas

6.2.3. Coste total

El coste total del proyecto se calculará en base a los costes materiales y personales, más el margen de seguridad por posibles problemas, más el beneficio y por último el I.V.A.

Concepto	***Coste***
Coste de material	254,00 €
Coste de personal	9.746,00 €
Margen de seguridad (15%)	1500,00 €
Beneficio (35%)	4.025,00 €
I.V.A	3.260,25 €
Total	18.785,25 €

Tabla 33 Coste total

El coste total del proyecto será de 18.785,25 €.

6.3. Entorno socio-económico

Este proyecto no tiene como objetivo generar dinero con el trabajo realizado, ya que este proyecto aún está en desarrollo por lo explicado anteriormente en el apartado *5 Marco Legal* a parte que todavía esta tecnología no está completa para un funcionamiento correcto.

El proyecto que se ha realizado mejora la actual solución y también ayuda a la iniciación de nuevos proyectos que mejoren la actual solución. Después de acabar el proyecto se podrá empezar a tener una remuneración económica del proyecto, aunque también depende de otros factores como el sector legal.

Cuando el proyecto esté terminado como se ha explicado anteriormente, el piloto automático de los drones tiene aplicaciones muy diferentes. A través de una cámara, el sistema de piloto automático puede vigilar una zona de forma continua y enviar la información de la cámara a un sistema de reconocimiento para encontrar problemas de diferente tipo, por ejemplo, reconocimiento facial, reconocimiento de enfermedades en cultivos, vigilancia, seguridad, etc.

Hay varias utilidades que se han nombrado anteriormente y que no tienen una remuneración económica directa, pero mejora la eficiencia del sistema, por ejemplo, en seguridad vial. También se obtiene beneficios de forma directa como por ejemplo sustituyendo el dron en tareas de vigilancia donde actualmente haya varias personas dedicándose a esta tarea. Otra posible remuneración puede ser en la recogida de información en cultivos para mejorar la producción mediante técnicas de ayuda a la toma de decisiones.

7. CONCLUSIONES Y TRABAJOS FUTUROS

En este último apartado se elaborarán las conclusiones extraídas tras la realización del proyecto actual. También se ha decido elaborar un apartado de trabajos futuros donde se explicarán posibles trabajos que mejorarían el proyecto actual o el proyecto global donde se engloba este proyecto.

7.1. Conclusiones

Este proyecto se ha desarrollado con el fin de mejorar el actual sistema de piloto automático implementado por DroneCode. Como DroneCode es un sistema en desarrollo se analizó el sistema para encontrar un proyecto que mejorara el mismo y aportara una carga de aprendizaje necesaria para la realización del trabajo. Después de analizar el sistema se decidió realizar un proyecto sencillo donde el objetivo principal es el análisis y evaluación del sistema DroneCode para que en posteriores proyectos puedan realizar una mejora más completa sin tener que realizar la tarea ya realizada en este proyecto.

Durante el actual proyecto se ha realizado un análisis completo del sistema PX4 y posibles soluciones para interactuar con el sistema, al ser PX4 un sistema de código abierto la documentación que en un principio es muy extensa, cuando nos acercamos a los detalles del desarrollo no hay un manual claro donde se explique como desarrollar, solo hay unas pequeñas guías de como empezar a usar algunos sistemas como uORB o de como crear una tarea secundaria que se ejecute en segundo plano y realice una tarea completa.

Por los problemas encontrados a la hora de desarrollar el sistema el proyecto ha durado más de lo esperado y se ha tenido que investigar cómo funcionan otros módulos del sistema y cómo utilizan de forma completa todos los componentes de PX4.

Después de la realización de este proyecto se ha comprobado que el sistema PX4 aún necesita mucho desarrollo y hay muchas alternativas para continuar trabajando en el sistema, como la implementación de un driver para el sensor de distancia, la mejora del sistema de evasión con análisis por inteligencia artificial, realización de pruebas reales con este sistema de evasión, etc.

7.2. Trabajos futuros

Como PX4 es un proyecto actualmente en desarrollo hay muchas mejoras que se pueden implementar para conseguir un sistema de vuelo autónomo que funcione correctamente y de forma totalmente autónoma sin la necesidad de que nadie vigile de forma exhaustiva el dron.

Este proyecto se ha centrado en abrir una nueva rama de desarrollo donde se podrán realizar proyectos que mejoren la funcionalidad de la realización de misiones autónomas con una mayor efectividad. En concreto este proyecto se puede mejorar añadiendo un controlador (driver) para obtener los datos del sensor de distancia laser e incluirlo en el mensaje de comunicación donde se lee en el sistema. También se podría realizar un proyecto con otro tipo de sensor de distancia que no sea laser como un proyecto que actualmente está en desarrollo que sería por ejemplo un radar.

También es necesario la implementación de un sistema de evasión más preciso y que funcione de una forma más efectiva con técnicas de inteligencia artificial, este sistema puede que necesite una capacidad de cómputo mayor a la proporcionada por Pixhawk por lo que se tendría que implementar en un computador paralelo que se comunique con el sistema principal a través de Mavlink pero con el mismo esquema estructural.

Otro proyecto que se puede desarrollar es analizar la computación dentro de Pixhawk para realizar pruebas dentro del hardware donde posteriormente se realizarán los vuelos reales. Para este proyecto es necesario que el proyecto de crear el controlador esté finalizado ya que es imprescindible a la hora de usar el sistema con un sensor real.

También este proyecto puede servir de guía para abrir nuevas líneas de desarrollo donde se generen nuevos componentes que mejoren el sistema PX4 como por ejemplo un sistema de comandos para realizar diferentes acciones directamente desde el ordenador ya que QGroundControl no implementa todos los comandos que soporta PX4.

8. BIBLIOGRAFÍA

[1] L. Meier, “First Application Tutorial,” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/apps/hello_sky.html#first-application-tutorial-hello-sky.

[2] “Vehículo aéreo no tripulado” 13-Sep-2018. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Vehículo_aéreo_no_tripulado - Aplicaciones

[3] Albatross, “History of C++” 03-Mar-2009. [En línea]. Disponible en: http://www.cplusplus.com/info/history/.

[4] L. Meier, “MAVLink Messaging” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/middleware/mavlink.html

[5] L. Meier, “PX4 Reference Flight Controller Design” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/debug/reference-design.html

[6] L. Meier, “SITL Simulation Environment” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/simulation/

[7] L. Meier, “jMAVSim with SITL” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/simulation/jmavsim.html.

[8] L. Meier, “Gazebo Simulation” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/simulation/gazebo.html.

[9] L. Meier, “PX4 Architectural Overview” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/concept/architecture.html.

[10] L. Meier, “uORB Messaging,” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/middleware/uorb.html

[11] L. Meier, “uORB Publication/Subscription Graph” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/middleware/uorb_graph.html

[12] L. Meier, “Flight Modes” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/concept/flight_modes.html

[13] L. Meier, “Offboard Control,” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/ros/offboard_control.html.

[14] M. Hollmann, “Radar” 2007. [En línea]. Disponible en: http://www.radarworld.org/huelsmeyer.html

[15] L. Meier, “Flight Log Analysis” 19-Jun-2018. [En línea]. Disponible en: https://dev.px4.io/en/log/flight_log_analysis.html.

[16] G. E. Nutt, “NuttX Real-Time Operating System” 2007. [En línea]. Disponible en: http://nuttx.org/.

[17] “Bases y tipos de cotización 2018” 2018. [Online]. Disponible en: http://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537.

I. SUMMARY

1. Introduction, motivation and objectives

In this chapter the project is presented. First, the motivation that led to the realization of the project will be described, then the objectives of the project will be described and finally the structure of this document will be explained.

Currently there is a high increase in automatic piloting systems, although there are still no complete autopilot systems as they depend on a pilot with the controls in hand for a vehicle to go in autopilot mode. Airplanes and cars are vehicles that already incorporate such systems as, for example, the Autopilot system of Tesla.

The use of autopilots in vehicles is a great revolution. From the point of view of road safety, the effective use of autopilots would save a large number of victims and road infrastructure since, mainly on the road, the vast majority of accidents are caused by a human factor. Applied to the drone, the autopilot would amplify its possibilities as a tool to access at difficult-to-access places for other vehicles, making its control easier, faster and more accessible to a wider group since its piloting is easy to learn and manage.

The tasks for which most drones are currently used, are recording work either in the private or professional sector, and for security work aimed at the professional sector. An example of safety work for drones is traffic surveillance. If autopilots would be developed for this type of work, they would be able to do it in a faster and more effective way, without the need of pilots who continuously monitor the drone. The drone could control traffic continuously and it could be equipped with an accident detection or maximum speed control system.

Transport companies like Amazon are also investigating home delivery through drones with an autopilot, wich is, in my opinion more difficult to carry out in the close future since the current law does not allow these flights without a pilot.

This project focuses on improving an autopilot system that is currently under development and whose missions can already be executed autonomously, focusing this

project on an obstacle avoidance system with a laser distance sensor. This type of autopilot in the future will be a tool that will incorporate most of the vehicles so it is of great interest to investigate with them and develop new components.

This project will focus on developing an obstacle evasion system that will improve the autopilot system that has already a sufficiently correct operation, adding to this a system for detection and evasion of obstacles. Although the objective is to develop an evasion system, the project will focus on analyzing the architecture and designing secondary tasks that could work alongside the autopilot.

Specifically, this work will focus on DroneCode, that is a set of projects that contain all the necessary tools for the autonomous piloting of unmanned aerial vehicles. Within this project, two of its proposals will be modified and they will be used without modifying the rest of them. PX4 and Gazebo will be the two DronCode projects to be modified, with PX4 being the flight controller that handles all the actions that the drone must perform for the autonomous flight. This project is one of the most important parts of DroneCode since it contains the autopilot system and the drone control system. On the other hand, Gazebo is a simulator specializing in robotics that will generate scenarios and vehicles with the sensors and actuators required to perform tests using software before using the drone in real tests.

2. PROJECT

Unmanned aerial vehicles (UAV) or commonly known as drones, are aircraft that do not need to be manned for proper operation.

The first drones emerged in the First World War as a target in training and as a defense against zeppelins, although they were vehicles that did not contain all the driving options of a common vehicle. Until the end of the 20th century, it was not possible to make vehicles with total autonomy.

Drones are currently very advanced and are used for a wide variety of applications ranging from recreational use to scientific research, rescue, border surveillance, use in commercial recording, etc. The most common uses are:

- In events: as football matches where the use of the drone allows aerial shots from angles that ordinary cameras are not able to access.
- In emergency situations: given that drones have a great capacity for maneuver it allows them to access places where human access or with other types of vehicles is very difficult or dangerous.

- Rural areas: for farmers these devices have a great potential as it allows access to farming areas where access with another vehicle could damage the crop and they can also go over several hectares in a fast way taking high quality photos or videos to control pests or find possible crop problems.

- Biological research: this research focuses on birds since the drone can reproduce migratory routes and allow researchers to collect information from these routes to carry out their studies.

- Recreational: this use is raising as drones are becoming more accessible over time, for fans of filmography or photography, drones are currently essential tools. There is also a modality that is becoming fashionable and they are the drone races or the acrobatic drones, these drones are usually small and very fast and they are controlled through an FPV.

The increasing supply and commercialization of these aircraft is causing the emergence of simpler models to handle and of cheaper construction, so it has become very accessible to the general population, using them personally as recording vehicles to take aerial shots

Currently it is beginning to develop drones that incorporate autopilot systems, although it is not a highly developed system yet. Current systems require human interaction or another system to mark the route that the drone has to follow, being the resolution of problems during the flight also responsibility of the human pilot. All this interaction is done through control stations. As these systems are still being developed, there are not many applications, being used for the moment for simple tasks such as, for example, for surveillance or treatment of agricultural land.

Automatic pilots are becoming more common, and companies dedicated to the manufacture and development of drones as DJI, have their own autopilot systems, although there are also open source projects that have very advanced autopilot systems such as, for example, DronCode or Ardupilot. These automatic pilots do not have

important utilities yet given that their development is not complete and the law still does not allow the flight of these completely autonomous vehicles.

DroneCode is a software, hardware and control solution for unmanned aerial vehicles (UAV). This system contains an autopilot system for the UAV, which works in conjunction with a control station and a computer that controls all the available hardware of the drone. The goal of DroneCode is to be a better, safer, easier to use and more flexible platform than others open sources or private projects. DroneCode works together with its own community and the Linux Foundation in order to expand itself.

DroneCode is formed by a set of open source projects or open hardware, these projects are:

- PX4: DroneCode flight control.
- QGroundControl: DroneCode control station.
- MAVLink: communication system between the control station and the vehicle.
- Pixhawk: hardware where flight control will be executed, it is also adapted to Qualcomm Snapdragon Flight and Intel Aero Ready.
- Gazebo: software simulator, also available JMavSim and AirSim.

PX4 is an open source project, the autopilot is made entirely in the C ++ programming language. This project contains a system of guidance, navigation and control of the algorithms for the autonomous flight system, being valid for unmanned multirotor vehicles and airplanes.

PX4 is a system created for different projects and although the aerial vehicle industry is still under development, there are already several cases of use for these systems, both for consumers and for industrial use. For example, PX4 has an option to follow a user through the mobile phone, this system can be used to make videos without having to drive the vehicle. There are also uses for transporting merchandise, for monitoring buildings or crops and many other uses that are yet to come.

PX4 is the system where most of this project will be housed since it is the system where the automatic pilot that controls the drone is located, and in this automatic pilot is where the event handling system will be integrated

Gazebo is an easy to use robotic simulator that allows you through a set of tools making a design quickly and easily which performs a series of tests to see if the algorithms work correctly. Gazebo includes a simulation of both outdoor and indoor environments. It also contains a powerful physical engine that allows accurate simulation. Gazebo also includes a set of tutorials that are available on its website to start using it in a simple way. Apart from all of this, Gazebo is free and has a large community that creates many models that we can use later.

Gazebo is going to be used as the simulator of this project, since it is the most complete simulator and that has a greater number of scenarios and vehicles than the simulators used in DroneCode. In later sections it will be explained in detail what has been used to carry out this project.

3. DEVELOPED PROPOSAL

As explained above, the main objective of the project is to analyze the structure of the DroneCode system, specifically how to continue developing inside PX4. This is a complicated task due to PX4 is a very complex system that includes many components and the development within a system where you do not know the internal functioning is very complicated. To develop a component where the analysis made above can be tested, an event handling system will be used, specifically an obstacle evasion system through a distance sensor. Next, an image that will help to understand the development made it is shown.

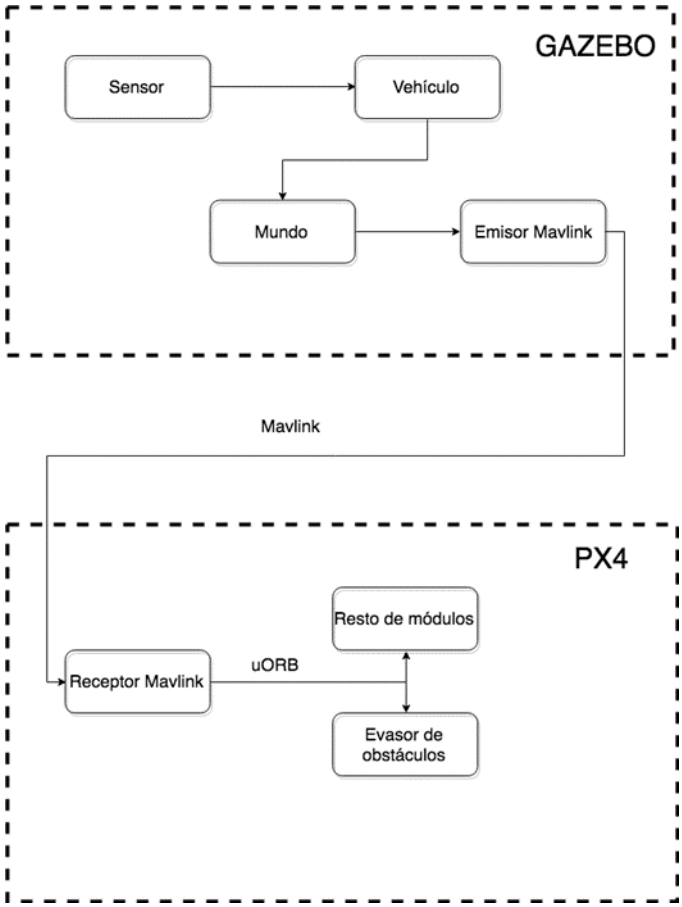


Imagen 36 Esquema del nuevo componente

As we can see in the *Imagen 36*, we must add the sensor module within the simulator, which is part of the vehicle, and in turn, the vehicle is part of a world in which we have all the modules of the simulation scenario and other modules such as communication through Mavlink, which is an external communication system. Since the communication module and the sensor module are in the same world, they can communicate with each other to send the information to PX4 via Mavlink with the distance sensor message already predefined.

The reception of the message in PX4 of the distance sensor is already implemented, so we must subscribe to the message of the distance sensor to obtain the data sent from the simulator. To execute the evasion actions, the internal communication system with the command message will also be used.

After obtaining the angle and the distance where the obstacle is located, it is encapsulated in the message predefined by Mavlink and the information is sent to PX4. After analyzing the information received in PX4 it was possible to verify that the frequency that Mavlink works with, it is not enough to send the distance sensor information correctly.

It could be checked that modifying the speed of the uper part of the sensor it achieved a frequency of about eighty degrees in each reading. To solve this problem it was necessary to sacrifice accuracy in the sensor, but the reader is still just as safe sending important information. For this, we used the following system:

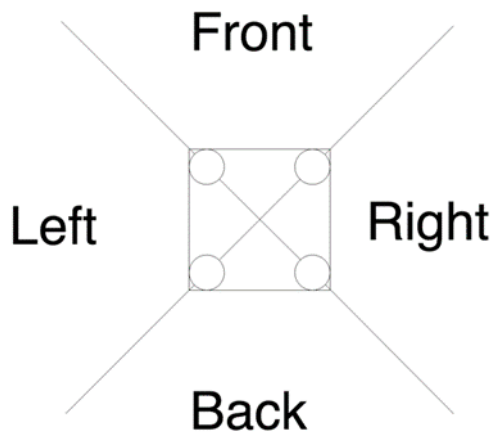


Imagen 37 Esquema problema de latencia

As we can see in *Imagen 37*, information of the sensor is sent every ninety degrees, in this case the minimum distance obtained in the corresponding interval is sent. As we explained above, the information is sent with a frequency of eighty degrees, so every nine shipments, the sensor will repeat information, which is not a problem. This system will not have as great precision as if all the angles with their information will be sent, but it can still be detected quite effectively if there is a danger in the direction the vehicle is moving.

To make a functional detection system, although it is not the main objective of this project, it will be done a detection system composed by three main phases: No danger, danger and evasion. Next, each of the phases will be described.

- No danger: in this phase only the sensor's distance will be checked and compared with the safety distance. If the sensor's distance is smaller, then the system will adopt the danger phase.

- Danger: this phase is created to avoid false readings since the probability of two false readings is very low. What is probable and common is that the distance read by the sensor is the distance of the drone to the ground and therefore an error.

- Evasion: in this phase the drone will execute the evasion maneuver to the side where the laser sensor does not detect any danger during the evasion.

4. CONCLUSIONS AND FUTURE PROYECTS

This project has been developed in order to improve the current autopilot system implemented by DroneCode. Since DroneCode is a system under development, the system was analyzed to find a project that would improve the system and provide a learning load necessary for the realization of the study. After analyzing the system, it was decided to carry out a simple project where the main objective was the analysis and evaluation of the DroneCode system so that in later projects they can make a more complete improvement without having to carry out the task already done in this project.

During the current project, a complete analysis of the PX4 system and possible solutions to interact with the system has been made, given that PX4 is an open source system, the documentation that at first is very extensive, when we approach the details of the development there is no a clear manual explaining how to develop, there are only some short guides about how to start using some systems like uORB or how to create a secondary task that runs in the background and that performs a complete task.

Due to the problems found when we were developing the system, the project has lasted longer than expected and it has been necessary to investigate how other system modules work and how they use all the PX4 components in a complete way.

After the realization of this project it has been verified that the PX4 system still needs a lot of development and there are many alternatives to continue working on the system, such as the implementation of a driver for the distance sensor, the improvement of the evasion system with analysis by artificial intelligence, real testing with this evasion system, etc.

As PX4 is a project currently under development, there are many improvements that can be implemented to achieve a self-contained flight system that works properly and completely autonomous without the need for anyone to monitor the drone exhaustively.

This project has focused on opening a new area of development where projects that improve the functionality of carrying out autonomous missions with greater effectivenes can be carried out. In particular, this project can be improved by adding a driver to obtain the laser distance sensor data and include it in the communication message where it is read in the system. It could also be done a project with another type of distance sensor that is not laser as a project that is currently under development, which would be, for example, a radar.

It is also necessary to implement a more precise evasion system that works more effectively with artificial intelligence techniques, this system may need a computing capacity greater than provided by Pixhawk, so it would have to be implemented in a parallel computer that communicates with the main system through Mavlink but with the same structural scheme.

Another project that can be developed is to analyze the computation within Pixhawk to perform tests inside the hardware where real flights will be made later. For this project it is necessary that the project of creating the controller is finalized since it is essential when using the system with a real sensor.

This project can also serve as a guide to open new lines of development where new components that improve the PX4 system are generated, such as a command system to perform different actions directly from the computer given that QGroundControl does not implement all the commands that PX4 supports.